Virginia
CENTER of
EXCELLENCE
for Software Reuse and Technology Transfer

January 31, 1995

Department of the Navy
Office of Naval Research Resident Representative
Georgia Institute of Technology
206 O'Keefe Building
Atlanta, GA 30332–0490

Reference: Grant MDA972–92–J–1018

Dear Sir:

In accordance with the referenced grant the Virginia Center of Excellence hereby submits the following special report documentation:

1. A Tailorable Process for Systems Engineering

   – Reference SPC–94095–CMC, Version 01.00.05, January 1995

This delivery is made for the subject grant under Section 9.1.b. If you have any questions, please contact me at (703) 742–7172 or Mr. Tim Powell at (703) 742–7264.

Sincerely,

Dennis Dwyer
Controller

/djk
Enclosure

cc: Project Office
Mr. John Foreman
ARPA/SISTO
801 North Randolph St., Suite 400
Arlington, VA 22203

Defense Technical Information Center
Attn: DTIC–OCC
Cameron Station, Bldg. 5
Alexandria, VA 22304–6145

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1995	Technical Report

4. TITLE AND SUBTITLE
A Tailorable Process for Systems Engineering

5. FUNDING NUMBERS
G MDA972-92-J-1018

6. AUTHOR(S)
Produced by Software Productivity Consortium under contract to Virginia Center of Excellence.

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Virginia Center of Excellence
SPC Building
2214 Rock Hill Road
Herndon, VA 22070

8. PERFORMING ORGANIZATION REPORT NUMBER
SPC-94095-CMC
Version 01.00.04

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ARPA/SISTO
801 N. Randolph St.
Suite 400
Arlington, VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE
1

13. ABSTRACT ***(Maximum 200 words)***

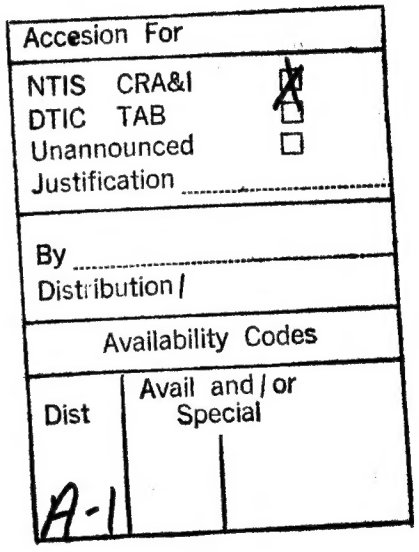
This report describes the Generic Systems Engineering Process (GSEP). The GSEP purpose is to provide systems engineers with the guidance and structure needed to efficiently develop high-quality, large, complex systems. To accomplish this, the GSEP defines an approach for communicating and coordinating an engineering effort, integrates the management and technical activities, incorporates risk management activities, and provides engineering guidance that is detailed enough to be helpful, yet sufficiently flexible to allow for creativity.

The GSEP is a generic development process and, as such, it can be applied to a wide variety of systems engineering efforts. The GSEP approach of defining a generic process and explaining how to create a specific process from the generic one, provides all the benefits of more traditional processes that are very specific in nature, but additionally provides the ability to customize the GSEP for specific applications.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Systems Engineering Process, Systems Engineering, Process Tailoring, Process Definition	250
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

A Tailorable Process for Systems Engineering



SPC-94095-CMC
Version 01.00.05

January 1995

A Tailorable Process for Systems Engineering

SPC-94095-CMC

Version 01.00.05

January 1995

Mike Cochran
Sue Rose
Grady Campbell
Linda Parker Gates
Kent A. Johnson
Chuck Lynch
Mary Skipp

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION
under contract to the
VIRGINIA CENTER OF EXCELLENCE
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER

SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

The tools mentioned in this report are not specifically recommended or evaluated, either among themselves or relative to other commercially available tools that are not mentioned. Furthermore, the accuracy and completeness of individual tool descriptions have not been validated with the vendors offering these tools and should not be considered authoritative. For such information, the respective vendors should be contacted directly.

CONTENTS

FIGURES

TABLES

This page intentionally left blank.

ACKNOWLEDGMENTS

The Software Productivity Consortium wishes to recognize the following contributors to the development of the Generic Systems Engineering Process (GSEP) technical report:

- The GSEP development team: Kirsten Blakemore, Mike Cochran, Sandy Friedenthal, Chuck Lynch, Susan Rose, Mary Skipp.
- Sandy Friedenthal, a Martin Marietta assignee, for sharing his systems engineering expertise and for his reviews, discussions, and advice in the creation of the GSEP and this report.
- Our internal reviewers: Richard Bechtold, Grady Campbell, Lisa Finneran, Kent A. Johnson, Rich McCabe, and Hal Pierson for their comments and suggestions.
- Kirsten Blakemore for managing most of the GSEP development effort.
- Tim Powell for his role as program manager and his review and corrections to the Systems Engineering Capabiltiy Maturity Model (SE-CMM) mapping in Appendix D.
- Dave Nettles for his support as project manager.
- Brian Mar, University of Washington, for reviewing and providing feedback.
- Dave Deitrich, Martin Marietta, Bethesda, for reviewing and providing feedback.
- Dinesh Verma, Virginia Tech, for reviewing and providing feedback.
- Donna Garfield for her support of the GSEP database creation and population.
- The Evolutionary Spiral Process (ESP) development team for creation of the ESP concepts and the initial version of the GSEP's management activity specifications.
- Grumman Aircraft Systems in Bethpage, New York (now Northrop Grumman) for their openness with their own systems engineering process. Specific thanks to Dick Edelmann who coordinated a 1-day meeting between the GSEP development team and his systems engineering team members: Cliff Beardsly, Hal Haverson, Alan Kocka, John Liney, Bob Lunes, Mike Sier, and Ron Tarpinean.
- The systems engineering Technical Advisory Group for helping us focus on areas of concern in systems engineering.
- The Consortium Visiting Committee, especially Wolt Fabrycky, Jerry Lake, and Brian Mar.

- John Brackett, Fred Hills, Jerry Pixton, and Sam Redwine for early input into the process and assisting in defining its scope.

- Phil Babel and John Kordick who gave a presentation that included a systems engineering process engine.

- Dennise Buede, George Mason University, for the presentation that he did on the leveling of systems engineering activities.

- Bobbie Troy for her technical editing and Debbie Morgan and Deborah Tipeni for their word processing assistance.

Appendix G represents a summary of the results of a project performed at George Mason University, with support from Virginia's Center for Innovative Technology under Contract INF-94-004. The project was conducted by Dr. John Brackett, Research Professor, and Dr. Andrew Sage, Professor, in the Department of Information Technology and Engineering, School of Information Technology and Engineering.

EXECUTIVE SUMMARY

This technical report describes the Generic Systems Engineering Process (GSEP) created by the Software Productivity Consortium (the Consortium). GSEP's purpose is to provide systems engineers with the guidance and structure needed to efficiently develop high-quality, large, complex systems. To accomplish this, the GSEP defines an approach for communicating and coordinating an engineering effort, integrates the management and technical activities, incorporates risk management activities, and provides engineering guidance that is detailed enough to be helpful, yet sufficiently flexible to allow for creativity.

Communication and Coordination

Although decomposition of large and/or complex systems into smaller, more manageable parts reduces the complexity of developing each part, it increases the necessity for communication between and coordination of the development efforts for each part. GSEP defines the communications links and the work product flows between activities and establishes the interfaces necessary to ensure successful integration of each level of system decomposition.

Management and Technical Activities

GSEP balances the integration of management and technical activities. This balance ensures that management is provided the technical expertise necessary for decision making and that the engineers are provided the plans and procedures for meeting customer, user, and organizational expectations. The GSEP management approach emphasizes commitment and buy-in by all system stakeholders (e.g., customer, user, managers, developers, maintainers). This affords stakeholders the opportunity to provide input to decision making and ensures that all stakeholder views are recognized.

Risk Management

In GSEP, risk management is an important part of the management and technical activities. GSEP is a risk-driven process. This means that risk analysis results are used in planning and decision making, leading to system-wide risk reduction. Reduction in management and technical risks increases the probability that a quality system will be delivered on time and within budget.

Separation of Concerns

GSEP's emphasis on defining objectives provides the means for achieving separation of concerns. An engineer separates concerns by compartmentalizing issues that must be dealt with when engineering a system. GSEP ensures that the objectives for performing the development activities are clearly defined. Objectives provide an understanding of what must be done and why, without overly constraining the creativity of the activity performers. Activity completion is based on accomplishing its defined objectives. This delineates the scope of each activity, making that activity's concerns separate from those of other activities. This reduces the probability that work products created by the activity will be used for purposes other than those for which they were created.

A Generic Process

The GSEP is a generic development process and, as such, it can be applied to a wide variety of systems engineering efforts. The GSEP approach of defining a generic process and tailoring the process to create a specific process from the generic one, provides all the benefits of more traditional processes that are very specific in nature. Additionally, this approach means that GSEP is scalable and can be applied to large, complex system development efforts and, yet, is also applicable to smaller projects. A high-level GSEP process is initially created that is customized to the project's specific needs. As more project process requirements are identified, an enactable GSEP process is created to address those requirements. This ensures that the GSEP process is always appropriate for the project's system development effort.

Engineering for Reuse

GSEP's flexibility enables its application for engineering reusable system parts. This report describes how the GSEP can be used to engineer reusable parts and how groups of similar systems can be engineered more efficiently by applying the GSEP to the creation of common pieces.

Process Improvement

The GSEP includes process improvement mechanisms. Lessons learned are continually documented and used to improve the quality of the delivered products. Process improvement can be realized in the following areas: in the systems engineering process; in better mechanisms for updating and evolving systems engineering processes; and in the methods, procedures, or organizational standards that are used to carry out the systems engineering process. As improvements are made, productivity and product quality increase.

1. INTRODUCTION

1.1 OVERVIEW

This technical report describes the Generic Systems Engineering Process (GSEP) created by the Software Productivity Consortium (the Consortium).The Consortium developed the GSEP to provide systems engineers with the guidance and structure needed to efficiently develop high-quality, large, complex systems. To achieve this, the GSEP deals with the complexity associated with developing large, multifaceted systems, including coordinating the effort and communicating pertinent information, integrating management and technical activities, and identifying and mitigating risks. GSEP provides systems engineering guidance that is detailed enough to be helpful, yet sufficiently flexible to allow for creativity.

Although other systems engineering processes exist, none meet all the objectives of the GSEP.

1.2 OBJECTIVES

There are two categories of objectives that GSEP was created to meet: objectives that lead to an improved ability for systems engineers to perform their work and objectives that help transfer the GSEP technology.

1.2.1 Systems Engineering Objectives

The main purpose behind GSEP is to improve the systems engineering process, allowing systems engineers to perform their work better. Thus, the Consortium created GSEP to meet the following objectives:

- ***Address and integrate the management and technical perspectives.*** Successful management requires technical insight and successful technical activities must be well managed. Both technical and managerial risks must be identified, analyzed, and mitigated. Communications mechanisms must be established to facilitate the flow of management and technical information.

- ***Provide scalability.*** Scalability ensures that the process is appropriate for use on any size systems engineering effort, from small to large, and ensures that the process is definable at a sufficiently high level that it can be generally applied while providing the means to derive a very detailed, enactable process.

- ***Incorporate process improvement mechanisms.*** As engineers use a process, they discover areas where the process could be improved. It is important for a systems engineering process to have a mechanism for incorporating this information into the organization's process improvement efforts.

- ***Integrate with existing processes.*** Investigation and application of existing systems engineering processes is preferable to reinventing or disregarding these efforts. Building on well-known and accepted systems engineering processes establishes a solid foundation for the new process. The fundamental concepts of the older, more familiar processes also help the new process gain acceptance.

- ***Support highly leveraged reuse.*** Reuse is necessary in today's systems engineering environment to provide:

  - Cost-effective systems development
  - Reduced development cycle time
  - Support for the trend toward reengineering and evolution of existing systems

1.2.2 Technology Transfer Objectives

The Consortium is sensitive to the problems that organizations face when adopting and institutionalizing processes and methods. Even the most beneficial process or method is not valuable if there is no plan for how it can be introduced into an organization. In order to help transfer the GSEP technology to organizations, GSEP was created to meet following objectives:

- ***Acceptable to a wide audience.*** If an organization is going to adopt a process, it must be appropriate for the type of systems engineering that organization does. In addition, many big organizations engineer a variety of systems, and having a single process that is applicable to all of them provides benefits such as reducing the amount of process training that is necessary to educate the staff.

- ***Customizable.*** To be useful, a process must be able to accommodate the needs and concerns of the projects to which it is applied.

- ***Well defined.*** A process should provide the detailed guidance that will help engineers successfully perform their work.

- ***Support partial adoption.*** Organizational change is not easy. Adopting a new process all at once can be disruptive and overwhelming to an organization. An evolutionary approach to process adoption can reduce the risk of failure by carefully introducing manageable change and then building on success.

1.3 GSEP'S APPROACH TO MEETING OBJECTIVES

GSEP's approach to meeting the systems engineering and technology transfer objectives is to provide an integrated, generic systems engineering process that addresses the objectives as a whole rather than addressing each objective independently.

- The GSEP provides a balanced process that equally emphasizes the management and technical perspectives. The GSEP defines the management and technical integration and includes mechanisms for communicating and coordinating information within the systems engineering process.

- The GSEP incorporates risk analysis activities and mechanisms into the systems engineering process.
- The GSEP supports the concept of concurrent engineering and integrated product teams (IPTs).
- The GSEP is scalable and can be applied to small and very large, complex systems. Through process tailoring, GSEP activities can be defined to be as general or as specific as necessary.
- The GSEP addresses process improvement by describing how process improvement is incorporated into the systems engineering process.
- Rather than ignore the systems engineering process work of others, GSEP was created to include and extend the work of others as much as possible.
- The GSEP is generic because it applies to the definition of any system. It does not specify the methods, tools, or techniques for performing the activities.
- The GSEP contains the activities necessary to customize the process to accommodate specific project needs and to create an enactable process that details individual tasks and responsibilities.
- The GSEP includes detailed activity descriptions, entrance and exit criteria, and work product descriptions to ensure it can be understood and applied.
- The GSEP can be phased into use in an organization through adoption in stages.

1.4 FOUNDATION FOR GSEP

Standards such as MIL-STD-499B (Department of Defense 1994) and P1220 (IEEE 1993) list requirements for a systems engineering process and describe concerns that must be addressed when tailoring the process. (The 499B standard referenced here is still in draft and is currently being converted to an industry standard, EIA Interim Standard 632.) Although the requirements and concerns are listed, the 499B standard does not describe how the systems engineering activities interact. This interaction is typically described in terms of the internal work products that are created in one activity and used in another. However, 499B emphasizes the production of external deliverables and provides little guidance on the creation of the internal work products. Conversely, the GSEP recognizes the importance of both internal work products and external deliverables and describes all activity outputs (regardless of whether their destination is internal or external) in detail. The GSEP and P1220/499B all use decomposition to deal with system complexity, but GSEP explains how the decomposed parts interact with each other, while 499B and P1220 do not.

Although different from 499B and P1220, GSEP is compliant with 499B and can be used to derive a process that maps to the P1220 standard.

Often, internal corporate standards deal with complexity by defining a process architecture that presumes a particular set of high-level subsystems (e.g., operational, manufacturing, etc.). For the most part, these subsystems depend on the type of system being built. Consequently, these process architectures are not sufficiently flexible to adapt to a wide variety of applications. Rather than

specifying a particular process architecture, the GSEP specifies how to create a process architecture by defining the subsystem processes and their interaction. Thus, the GSEP is capable of defining a process architecture similar to the ones defined in these standards, but, by decoupling the process architecture from the generic process and providing guidance on process architecting, the GSEP can achieve the same result with more flexibility.

Current systems engineering approaches address the systems engineering process from either a management or technical perspective. The management perspective focuses on the need to plan, control, monitor, and staff the engineering effort (Kerzner 1984). The intent of management is to know the status of the project relative to commitments it has made and to manipulate the resources and processes to allow the project to meet its stated company, customer, and user objectives. How this is accomplished is important because management must ensure an acceptable system, but the system being produced is not the focus of the management perspective. The technical perspective, on the other hand, focuses on the activities necessary to create a system that meets all of its technical requirements (Blanchard and Fabrycky 1990). The time frame in which the activities are carried out, who staffs the project, and the collection of status information, although important, are not the focus of the technical emphasis.

The management process can be defined in terms of the technical process, or vice versa. In his book, Blanchard (1991) discusses how to manage each of the technical activities, thus the management activities are viewed from a technical perspective. The GSEP defines a management process that takes a management perspective and a technical process that takes a technical perspective. These two processes are then integrated. Thus, each process is complete and neither process emphasizes nor constrains the other.

1.5 INTENDED AUDIENCE AND USES

The primary audience for this technical report consists of those professionals responsible for creating or modifying a systems engineering process for an organization. This audience typically includes process improvement group members and systems engineering professionals working on process definition activities. This audience will benefit by having a generic systems engineering process that they can use to derive or supplement their own process definition.

The secondary audience for this report consists of practicing systems engineers who can use the GSEP to help understand, articulate, and recommend changes to their current processes.

1.6 ORGANIZATION

This report includes the following sections and appendixes:

- **Section 1, Introduction.** This section describes the GSEP objectives, defines the primary and secondary audiences for the report, and provides an overview of each of the report sections.
- **Section 2, Systems Engineering Process Development Foundations.** This section describes the foundational concepts of systems engineering processes.
- **Section 3, Systems Engineering With GSEP.** This section summarizes the GSEP and describes how the GSEP addresses key foundational concepts.

- **Section 4, Applying Tailoring Concepts to GSEP.** This section describes general tailoring guidelines that are used to customize the GSEP to address specific project needs.

- **Section 5, GSEP-Based Engineering for Product Families.** This section defines a product family and briefly describes how the GSEP can be used to develop products efficiently through highly leveraged reuse.

- **Section 6, Conclusions**. This section summarizes the important GSEP concepts and describes the future plans for enhancement and application of GSEP.

Appendixes A through C define the detailed GSEP process description.

- **Appendix A, GSEP IDEF Diagrams.** This appendix contains the IDEF diagrams that define the GSEP.

- **Appendix B, GSEP Activity Descriptions.** This appendix contains the detailed activity descriptions for each of the GSEP activities.

- **Appendix C, GSEP Work Product Descriptions.** This appendix contains a description of each of the work products that are either inputs to or outputs from the GSEP activities.

Appendixes D and E describe the relationship of the GSEP to systems engineering standards and discuss the way that the GSEP was created with these standards in mind.

- **Appendix D, GSEP Mapping to Systems Engineering Standards.** This appendix provides a compliance mapping between the GSEP and both MIL-STD-499B and the Systems Engineering Capability Maturity Model (SE-CMM).

- **Appendix E, Engineering the GSEP.** This section describes how the GSEP was developed.

Appendixes F and G address the use of the GSEP in the fluid systems engineering environments that companies create to support internal systems engineering efforts.

- **Appendix F, GSEP Automated Support Environment.** This appendix defines a framework for an automated support environment for the GSEP. It supplies sources of information on systems engineering tools and surveys available tools that can be used when performing the GSEP activities.

- **Appendix G, Systems Engineering Trends Assessment.** This appendix describes the results of an analysis that was done to assess marketplace and technology trends and their potential impact on the practice of systems engineering.

This page intentionally left blank.

2. SYSTEMS ENGINEERING PROCESS DEVELOPMENT FOUNDATIONS

This section describes a systems engineering process, explains the foundational concepts upon which GSEP is based, and defines terms and concepts that will be used throughout this report.

2.1 DEFINING SYSTEMS ENGINEERING

The following definition is offered in order to establish the context for this report:

> Systems engineering is a comprehensive, iterative, problem solving approach for creating an optimized system solution to satisfy customer and user needs.

Systems engineering takes a system through an entire life cycle from concept exploration through disposal. Figure 1 shows typical life-cycle phases for a system during its life span and their associated stages delineated by milestones. Each milestone indicates a change in focus and defines the beginning and ending of a stage in the development of a system. At the same time there are always substages (e.g., the initial operating capability probably occurs early in the manufacturing stage). The brackets on the left of the diagram show the percentage of effort spent on the particular life-cycle phase at time 0 in the development of a system. As a system progresses through stages defined by milestones, the effort profile of relative phases changes accordingly. Completing stages by meeting key life-cycle milestones does not imply that the activities associated with a phase are complete, because the activities associated with a phase are likely to continue over the remaining life span of the system. As an example, development activities actually start in the concept exploration phase and are likely to continue until the system is disposed.

The development and concept exploration phases of the life cycle define the parts of the system to be developed (i.e., operational, manufacturing, support, etc.) and the processes for developing, implementing, and using each part. These phases do not include carrying out the processes for implementing and using the system parts. For example, when engineering a car, the system definition would describe the car system in sufficient detail to assemble the car, but the creation of the definition does not include the actual assembly of the car (which may be automated by a manufacturing system). In Figure 1, the arrows pointing to the development phase of the life cycle indicate that, during system development, the other life-cycle phases are defined, but not implemented.

Although some definitions limit the role of systems engineering to creation and allocation of system requirements, the definition of systems engineering in this technical report considers the systems engineering role to be broader and include the activities that are necessary to produce a system definition. In the car example, the system definition would include the architecture of the car and design for the subsystems (body, frame, etc.) and the architecture and design of software (cruise control), hardware (drive shaft), and procedures (repair manual, operating manual, etc.).

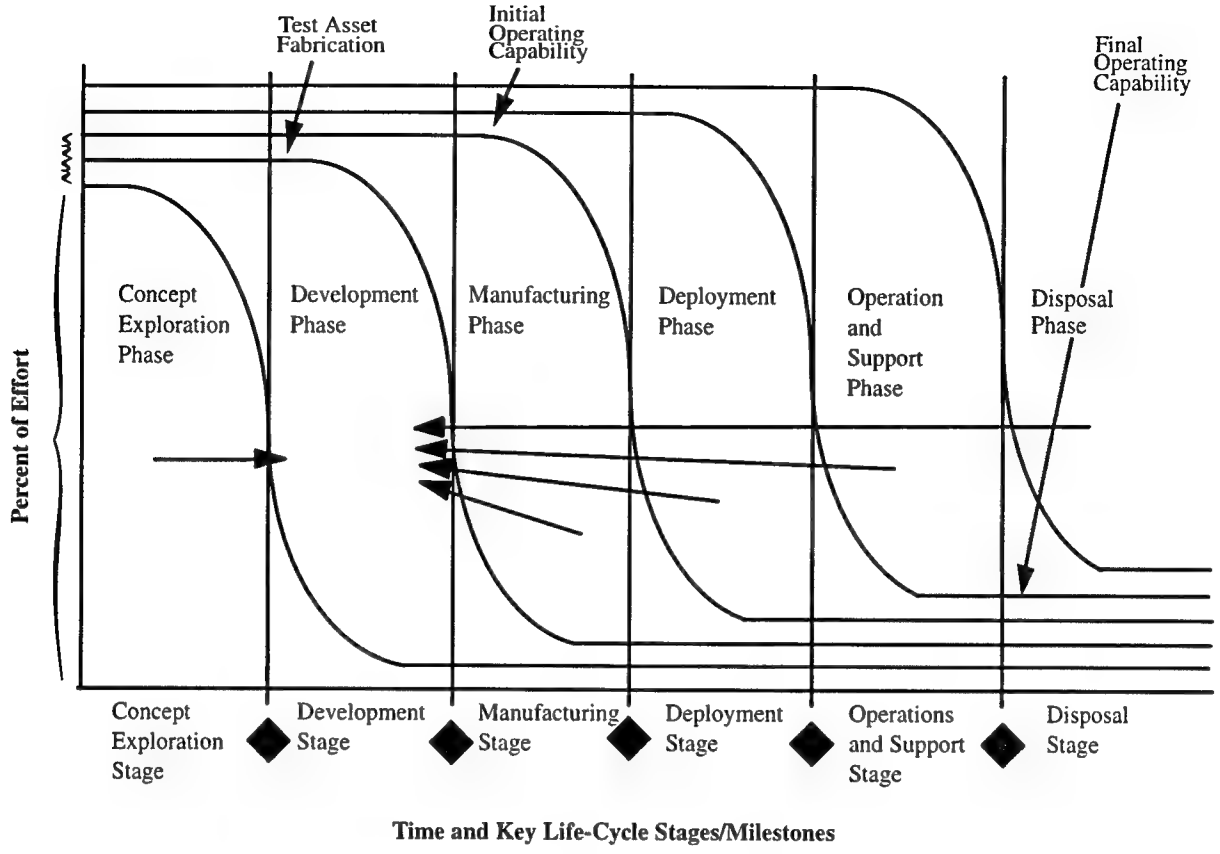


Figure 1. System Life-Cycle Phases

2.2 A PROCESS FOR SYSTEMS ENGINEERING

A **process** is a partially ordered set of steps intended to accomplish specified objectives. A **systems engineering process** provides engineering guidance for developing a system by defining what activities are necessary and their partial ordering, how information is exchanged between activities, and what work products must be produced.

A process must be comprehensive to be useful to a systems engineer responsible for following a particular systems engineering process. A process is considered comprehensive if it is:

- *Well Defined.* A process is considered well defined if it includes a description of each activity, the entrance and exit criteria for the activity, and the inputs and outputs for the activity. The process must also describe the activity ordering, the work product flow between activities, and controls and mechanisms associated with each activity. It is also desirable for the process to describe the metrics for assessing the work products that are produced and the roles necessary to carry out the activities.

- *Well Managed.* The process must define and describe the management activities and address management issues.

- *Scalable.* The process must be definable to be applicable to large and small systems.

- ***Disciplined.*** A process is disciplined if it defines mechanisms that can be used to ensure that the process is adequately followed.

2.2.1 Development Process Models

A development process model describes how work is performed over time. This is in contrast to life-cycle models that describe how a product matures over time. Figure 2 is a conceptual diagram of a life-cycle model. A life-cycle model defines a sequential set of product states. The Waterfall Process Model is an example of a life-cycle model (Royce 1970). The Waterfall describes the product states and their order (e.g., concept exploration stage, development stage, manufacturing stage, etc.). Life-cycle models, like the Waterfall, provide guidance on what must be done and the order in which it must be done, but not how to achieve each defined state. As a result of this perspective, a life-cycle model does not address, for example, how the continuing development process activities are performed and how they interact with manufacturing activities during the manufacturing stage.

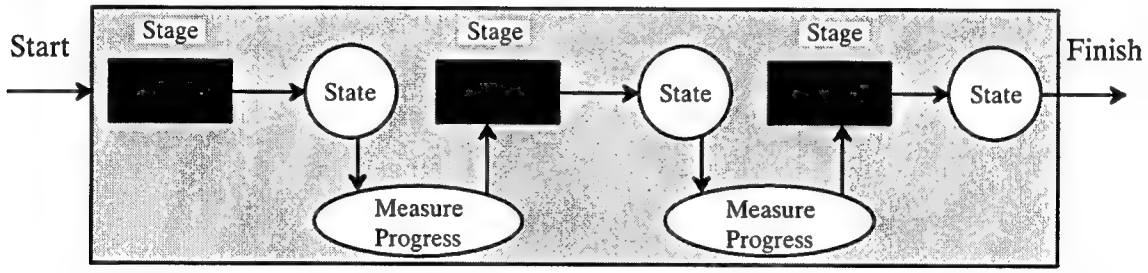


Figure 2. Life-Cycle Model

Development process models look at what is necessary to reach the product development states and define the processes for maturing the product so that it reaches the desired state. Figure 3 is a conceptual diagram of a development process model. Although Figure 2 is similar to Figure 3, life-cycle and development process models are substantially different because they have different focuses. The focus in life-cycle process models (Figure 2) is everything except what happens in the stages represented as "black boxes." The focus in development process models (Figure 3) is in only what happens in each of the stages, which are no longer black boxes. As a result of this perspective, a development process model addresses how the continuing development process activities are performed and how they interact with manufacturing activities during the manufacturing stage.

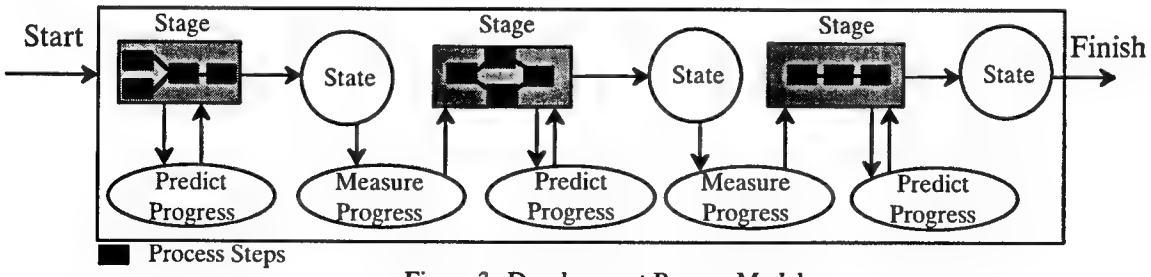


Figure 3. Development Process Model

In addition to defining how individual states are achieved, development process models also provide guidance on how the product moves between states. For example, development process models define the process for returning to previous states to make corrections, describe how iteration between product development steps is accomplished, and explain how products at various states are decomposed and later integrated.

The main difference between Figure 2 and 3 is that in Figure 3 it is possible to "see" into the stages, where in Figure 2 it is not. This is because development process models provide visibility into each stage. Visibility leads to repeatability because the more that is understood about what is being done in a stage to get to a desired state, the easier it is to duplicate the actions and reach the same state again. Repeatability leads to predictability because each repetition provides insight into the actions expected result. Thus, Figure 3 includes a Predict Progress activity that shows information coming out and going into each stage; there no similar information flow in the life-cycle process model in Figure 2. If a sequence of activities is predictable, then the activities can be improved. The progression from visibility to repeatability to predictability is the basis for process improvement.

Process visibility benefits managers because it provides a means to predict and control the product development stages. Process visibility benefits engineers because it provides them with the detailed guidance they need to reach the desired state.

Process visibility leads to the ability of development process models to exchange information during product stages rather than limiting information exchange to the product state milestones. That is because the more that is understood about the detailed activities being performed in every stage, the easier it is to understand what information a stage needs and in what other stage it is being produced.

2.2.2 Definition of a Generic Process

Development processes are on a continuum from generic to specific. A specific development process details the activities in a stage that are required to achieve a particular product state. In a specific development process, the process for each stage is unique to that stage and potentially different from the processes in other stages. On the other hand, in a generic development process, the same set of activities is used for each stage.

The generic development process is a more flexible approach for defining the product stages because it allows the development process to be independent of product state and, thus, independent of life cycle. If the life-cycle and development processes are independent, the same generic process can be used with whatever life cycle is most appropriate for the project. There are several benefits associated with generic development processes:

- ***Increased Familiarity With the Process.*** Because the same process is used repeatedly, practitioners become familiar with the process and need not be retrained for each project.
- ***Continuous Process Improvement.*** Repeatedly applying the same process allows for continuous process improvement. As the process is used, insight is gained into the areas where the process is weak and corrective action can be taken to improve the process.
- ***Organizational Process Standardization.*** A generic process promotes organizational process improvement through standardization of a process that does not change with each application to a different life cycle.

2.2.3 Process Tailoring

A generic process has a potential disadvantage in that the process may be so high-level that it does not supply the detailed guidance necessary to ensure successful development of each product state. To provide the required detailed guidance, the generic process must be made more specific. **Process tailoring** is defined as creating a more specific process from a more general one.

Process tailoring is done at different levels of detail. At the highest level, a generic process provides the basis for the process that will be used to create the system. In the Automobile System example (Figure 5), that tailoring consists of understanding the special process requirements for developing an automobile system and how the automobile system differs from other similar systems. Process tailoring is an important part of process engineering and is described in more detail in Section 4.

2.3 FUNDAMENTAL SYSTEMS ENGINEERING CONCEPTS

An understanding of the fundamental systems engineering concepts is important to fully appreciate the GSEP. A systems engineering process must address the following fundamental systems engineering concepts: integration of management and technical processes, dealing with system complexity, separation of concerns, support for concurrent engineering, and systems architectures.

2.3.1 INTEGRATION OF MANAGEMENT AND TECHNICAL PROCESSES

Research done at the Consortium makes it clear that the systems engineering community is polarized by those who view the systems engineering process as a management process and those who view it as a technical process.

Those who view systems engineering as a management process emphasize the management aspects using techniques such as total quality management and are concerned with techniques that specify how to:

- Perform the initial up-front planning at the start of the project
- Assess the project status
- Control the project during the system development life cycle

Those who view systems engineering as a technical process emphasize the technical development of the system using system specification and modeling techniques that allow insight into the system being built. This community is concerned with techniques that specify how to:

- Capture system requirements design and implementation
- Evaluate the system requirements design and implementation
- Analyze trade-offs in system design choices

A management process can be defined in terms of the technical process, or vice versa. If an engineer determined that the technical activities were the key activities, then a management plan could be put in place to manage each separate technical activity, and some higher level management activity would be put in place to coordinate the management of the different technical activities. If a manager determined that the management activities were the key activities, then the technical activities would be structured to fulfill each management objective.

An **integrated systems engineering process** defines a management subprocess that takes a management perspective and a technical subprocess that takes a technical perspective. Thus, the systems engineering process is complete, and neither subprocess emphasizes or constrains the other.

Differing views of systems engineering are found in a variety of systems engineering texts (Goode and Machol 1957); (Hall 1962); (Chestnut 1965, 1967); (Chase 1974); (Sage 1977); (Blanchard and Fabrycky 1981, 1990). More recently, the methods, tools, techniques, activities, functions, and purpose of systems engineering have been refined (Eisner 1988); (Blanchard 1991); (Rechtin 1991); (Chapman, Bayhill, and Wymore 1992); (Lacy 1992); (Sage 1992). The Department of Defense (DoD) has produced systems engineering standards MIL-STD-499A (Department of Defense 1974) and 499B (Department of Defense 1994). The National Council on Systems Engineering (NCOSE) was founded in 1991 to bring the systems engineering community together and has initiated working groups to address various aspects of systems engineering (Pohlmann 1993).

2.3.2 Dealing With System Complexity

To deal with complexity, a systems engineer has two tools: **system decomposition** and **system increments**. The engineer can use system decomposition to divide a large and/or complex system into smaller parts. Each part is a product to be produced and is considered a system in its own right. System increments, on the other hand, define the product states that are needed and each life-cycle milestone.

2.3.2.1 System Decomposition

System Structure. In system decomposition, the **system** is divided into **subsystems**. Perspective determines the difference between a system and a subsystem. To the engineer who must manage and/or create the system, its constituent parts are the subsystems, but to the engineer who must manage and/or create the subsystem, that subsystem is the system. The one distinguishing feature between a process to define a system and a process to define a subsystem is that a subsystem process must pass information back up to the system process, as well as down to other subsystem processes. A system process only passes information down to its subsystem processes.

At some point in the decomposition, the subsystem is no longer divisible into subsystems and contains only **components**. A component is defined as a portion of the subsystem (or system) that contains only basic **elements**. Elements are created externally (i.e., elements are created by some other system) and assembled to create the component. As a result, there is a process defined for creating the components, and decisions made during that process must be communicated to higher level processes. Thus, a component process passes information in one direction, to the subsystem processes above it. Since elements are created externally, there is no process defined for an element and, therefore, there is no process information to communicate. Figure 4 is a diagram showing the relationships between system, subsystems, components, and elements.

Note that in Figure 4 the solid lines that connect the system, subsystem, and component boxes indicate both a "contains" relationship and the transfer of process information between corresponding processes. Thus, a system may contain subsystems and components, and both subsystem and component processes pass information back up to the next higher level process. Also note that both system and subsystem processes pass information down to the next lower level process. The dashed lines in Figure 4 that connect components and elements indicate a "contains" relationship, but not a transfer of process information between components and elements.

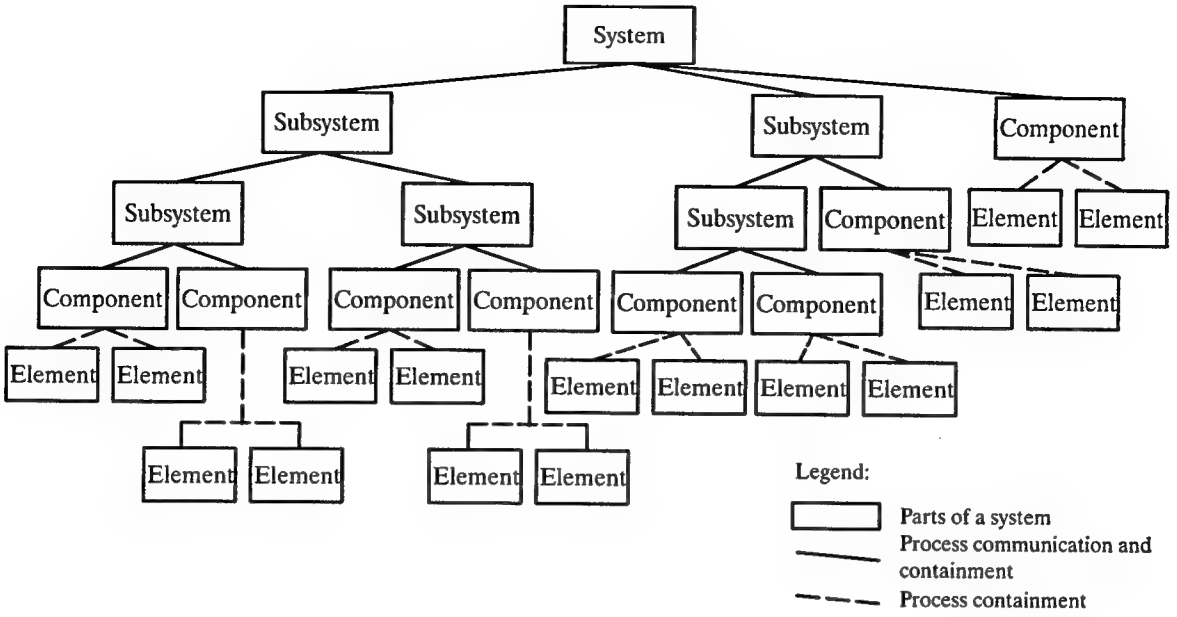


Figure 4. System, Subsystem, Component, Element Structure

Example of System Decomposition. To help communicate these concepts, this report includes an Automobile System example (see Figure 5). The Automobile System is composed of several subsystems, one of which is the Car Subsystem. The Car Subsystem is composed of the Body, Braking, and Powertrain Subsystems, and the Frame Component. The other Automobile subsystems, Driving Procedures, Maintenance, Manufacturing, Distribution, Marketing, and Recycling, are also composed of other subsystems, but they are not elaborated in Figure 5. In Figure 5, only the Powertrain Subsystem of the Car Subsystem is elaborated into its subsystems; they are the Engine Subsystem and the Drive Shaft and Transmission Components. An example element might be the bearings that are used in the u-joint that is part of the drive shaft.

Life-Cycle Phases. It is important to understand the mapping between general product states, in Figures 2 and 3, and systems, subsystems, and components, in Figure 4. Every system, subsystem, and component has a set of product states through which it progresses. The states are life cycle dependent, not system decomposition dependent. For example, a system may go through the requirements state, design state, development state, testing state, and release state, on its way to final delivery. The system might also go through a different, but equally valid set of states, like the development state, the delivery state, the trial usage state, the redesign state, the modify system state, the redelivery state, followed by another trail usage state and repeated update states. These are two different life-cycle models, but both are equally valid depictions of the possible development states. All of the system parts (i.e., systems, subsystems, and components) go through a set of development states, but even within the same system decomposition (i.e., two different subsystems within the same system), the states a particular system part passes through may be different from another system part. Consequently, the decomposition of the system into its constituent parts is independent of the life-cycle states that each of the system parts must go through.

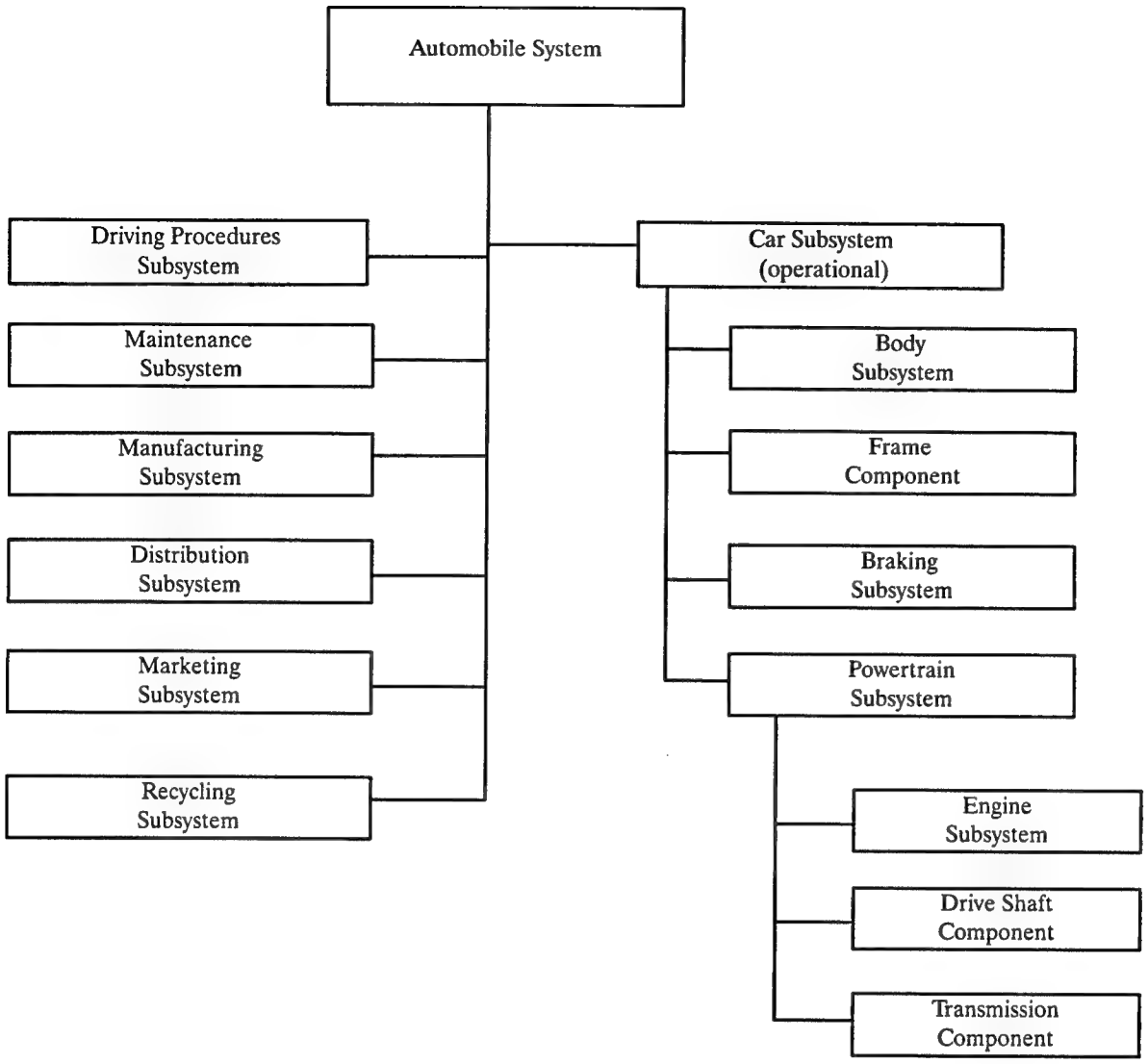


Figure 5. Automobile System Decomposition

The size of the system being created has no real impact on the above definitions of system decomposition and system parts. Thus, systems can be large or small because the distinction between a system and a subsystem is based on the direction of information flow between their processes, not their size. This is an important concept since, it is not the system structure that is important in a development process, it is how the system parts are developed and how communication and coordination are handled between the processes defining the parts. This concept goes back to the type of definition of a development process model. A development process model concerns itself with how the product states are achieved, not what the states are. Using the Automobile System example, the model's scope of concern is how the system is created. The model defines how each of the automobile subsystems is created and how they are combined to create the Automobile System. If the Automobile System is decomposed differently (e.g., Front End Subsystem, Rear End Subsystem, and Passenger Compartment Subsystem) the same development process model is still appropriate for its development.

Process Information Flow. It is equally important that information be allowed to flow between different subsystem/component processes as well as up and down the structure. For example, in the Automobile System, information flow between the Car Subsystem process and the Marketing Subsystem process is important in that an appropriate marketing strategy is developed and that the car that is built fulfills the Marketing Subsystem's promises. Also, there are many decisions (e.g., architecture and structural design decisions) that impact both the Body Subsystem and Frame Component, and these decisions must be adequately coordinated and communicated. Although it is possible for this type of information exchange, called "across" information flow, to follow the standard up and down information flow mechanisms, that approach would be inefficient and error prone. For example, if design decision alternatives and rationale for the automobile's body needed to go from the Body Subsystem up to the Car Subsystem and from there to the Automobile System and down to the Marketing Subsystem, and questions and comments from the Marketing Subsystem have to take the reverse path to get to the Body Subsystem, the communications route would slow progress and each communications exchange would introduce the possibility of erroneous information exchange (i.e., a real life telephone game example).

2.3.2.2 System Increments

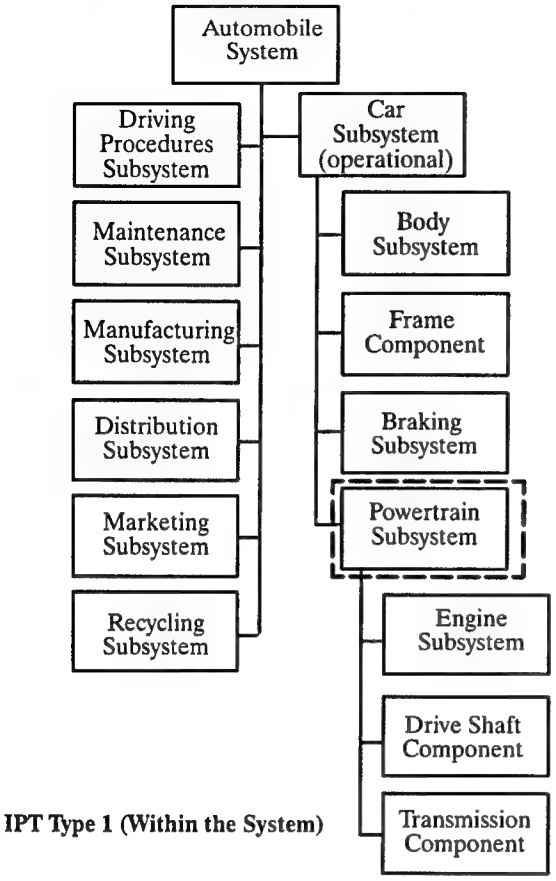
In addition to system decomposition, the systems engineer can deal with complexity by defining the incremental stages, or states, through which the system must transition on its way to completion. Dividing large and/or complex system development into product states and development stages is done by systems engineers and process engineers at the beginning of the project.

An increment defines the approach over time to develop the system. For example, an increment definition for the Braking Subsystem may include concept definition, conceptual prototype, working prototype, and final system definition for the Braking Subsystem. A different increment definition for the Braking Subsystem might be as simple as build 1, build 2, build 3, and build 4; where each build adds functionality or matures the Braking Subsystem toward its final system definition. Although the increment definition describes the states for the system parts, the increment definitions for the system parts are independent. For example, the Powertrain Subsystem increments are independent, though compatible, with the increments of its constituent parts (i.e., Engine Subsystem, Drive Shaft Component, and Transmission Component).

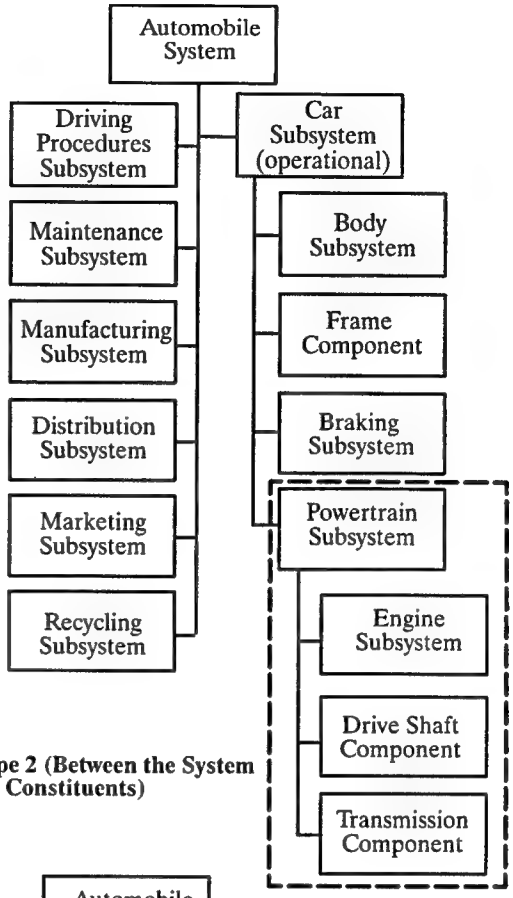
2.3.3 Support for Concurrent Engineering

As a result of the system decomposition necessary to deal with system complexity, communication and coordination in the systems engineering process are complicated. An IPT is a means to ensure communication and coordination of information that supports concurrent engineering. Concurrent engineering supports consideration of multiple viewpoints of engineering disciplines at the same time. In a systems engineering process, there are three types of IPTs (see Figure 6).

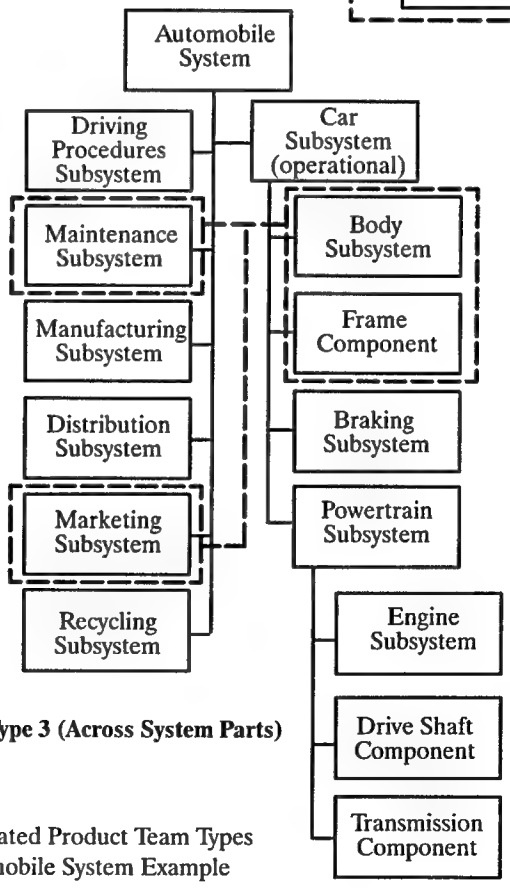
- ***IPT Type 1: Within the System.*** There is a type of IPT that engineers a system part (i.e., a system, subsystem, or component). The objective of this type of IPT is to ensure the internal coordination and exchange of information. That is, this type of IPT provides understanding and insight into the all of the work being performed to define the system part, not considering peer or constituent systems engineers inputs. An example of this type of IPT is the IPT in the Automobile System example (Figure 5) that is responsible for the Powertrain Subsystem. All members of the IPT would be responsible to bring their expertise in "ilities" (e.g., reliability or maintainability) to bear on the powertrain development.



IPT Type 1 (Within the System)



IPT Type 2 (Between the System and Its Constituents)



IPT Type 3 (Across System Parts)

Legend:

Shows source of IPT resources.

Figure 6. Integrated Product Team Types for the Automobile System Example

- ***IPT Type 2: Between the System and Its Constituents.*** This type of IPT provides coordination and communication between a system process and its constituent processes. The objective of this type of IPT is to ensure that the engineers defining a system communicate and coordinate with the engineers defining the constituent parts. An example of this type of IPT is the IPT in the Automobile System example (Figure 5) for the Powertrain Subsystem that contains members from the Powertrain Subsystem, the Engine Subsystem, and the Drive Shaft and Transmission Components. All members of the IPT would be responsible to bring their expertise in subsystem/component designs to bear on the powertrain development.

- ***IPT Type 3: Across System Parts.*** This type of IPT provides coordination and communication among processes defining system parts that require "across" information flow. The objective of this type of IPT is to provide the efficient transfer of ideas and decisions when needed. An example of this type of IPT in the Automobile System example (Figure 5) is the IPT that has members from the Marketing, Body, and Maintenance Subsystems and Frame Component whose objective is to ensure that the automobile is designed to meet marketing objectives, including an attractive body, a frame that meets safety requirements, and low maintenance costs. This type of IPT is the most difficult to create because:

  - Each system is unique and requires a unique set of the information exchange objectives.
  - These objectives typically cross structural boundaries and may be difficult to identify.
  - Teams that include members from different parts of the organizational structure may be difficult to form.

  However, this type of IPT is very important because it provides insights and understandings that impact the entire system.

Although there are three distinct types of IPTs, when IPTs are implemented, one or two IPTs may fill the objectives of all three types. The reason this report makes a distinction between the types is not to imply that each type must be separately implemented, but rather to ensure the objectives of all three types are considered when creating IPTs and that no important communication or coordination mechanism is overlooked.

2.3.4 Separation of Concerns

A useful development process is based on the fundamental concept that there are separate concerns that can be dealt with somewhat independently in different activities. These concerns impact the engineering decisions that are made while performing an activity. These decisions affect the work product(s) that the activity produces. The goal is to minimally constrain other activities that consider other concerns. For example, the engineer concentrating on requirements concerns is unable to consider all of the design concerns for large and/or complex systems at the same time and, thus, separates the requirements and design concerns.

During the requirements definition, engineers should not impose requirements on the design that are not absolutely necessary (i.e., overly constrain the design) because these restrictions may arbitrarily limit the design choices. This guideline dictates that it would be inappropriate for design decisions to be based solely on requirements allocations (e.g., the requirements decomposition mandating

allocation decisions). Although this concept may seem obvious, frequently work products created earlier in the engineering effort to satisfy specific objectives (e.g., requirements) are used later to meet other objectives (e.g., design) without sufficient analysis as to whether the work product adequately meets its current objectives. As an example, it may be reasonable for an automobile requirements specification to include aspects of braking requirements in the Frame Component and other aspects in the Powertrian Subsystem. It would not be reasonable for these requirements to constrain the design in such a way that a separate brake subsystem was not considered because of the requirements decomposition.

One particularly problematic place where separation of concerns should be applied is in the use of the high-level system functions. The statement of needs is used to create a set of high-level functions. The objective of these functions is to generate the initial set of requirements. The objective of the initial requirements is to provide the basis for additional decomposition of the requirements until a set of detailed requirements is generated. The objective of the detailed requirements is to specify the behavioral and performance requirements the design must meet. However, typically, the statement of needs is used to create the high-level functions, which are decomposed into the initial set of requirements, which are further and further decomposed until a design emerges from the detailed requirements constraints. Not only does this approach overly restrict the design, it incorrectly presumes that one objective of the high-level functions is to create a structure that would be used for the design and that one objective of the detailed requirements is to constrain the design.

2.3.5 Systems Architectures

When a system is engineered, the system architecture defines the structural parts of the system, how the parts are connected, and the interface with the external environment. The system architecture defines the subsystems, components, and elements that define the system and the interfaces between them. The system architecture represents the design and, as a result, constrains the design of the subsystems and components. The system architecture will also determine the level of reuse that can be practiced in that system.

When defining alternative system designs, an engineer should ensure that different architectures are considered rather than only defining minor differences in the subsystems or components that make up the architecture. If this is not done, then the various alternatives considered will not adequately explore the design space, and a poor system design may result. For precedented systems, there may be a set of architectures that have been used in the past that form a starting set. For unprecedented systems, innovation is required.

2.4 Engineering Product Families

A generic development process model is designed to successfully engineer a system and, certainly, that is its main purpose. But, in today's competitive market, successfully engineering a single development effort may not be sufficient to guarantee that the enterprise's engineering program is successful. One goal of a successful engineering program is that all of the enterprise's engineering efforts are coordinated such that the entire program is more efficient. In order to do this, an organization must plan to develop components, subsystems, and even systems that can be systematically reused by predicting future market and engineering trends. Although this is an ambitious undertaking, work is being done to provide insight into how this type of coordinated engineering can be accomplished. Product line-engineering, or the development of product families, is one engineering concept that provides guidance on how to efficiently engineer groups of similar systems.

The following terms are used in describing the concepts in engineering product families:

A **family of systems** is a set of systems that solve similar problems in similar ways.

A **product family** is a formalization of a family of systems and associated work products based on the similarities and variabilities among those problems and solutions.

Product family engineering is the development and evolution of a product family to facilitate the development of systems that solve similar specific problems. Thus, a product family is based on systems that have a set of common characteristics. Section 5 describes applying GSEP to product family engineering.

The concept of engineering product families has four primary distinguishing features:

- Formalization of a domain as a family of systems that share many common features, but that also vary in well-defined ways
- System building reduced to resolution of requirements and engineering decisions, representing the variations characteristic of a domain
- Reuse of artifacts through mechanical adaptation of domain engineering work products to satisfy requirements and engineering decisions
- Model-based analyses of described systems to help understand the implications of system-building decisions and to evaluate alternatives

The concept of a domain arose from a perception that organizations repeatedly build similar systems. To the degree that similarities can be identified and standardized, there is leverage to avoid redoing similar work in the future. However, by also identifying and standardizing essential variabilities in similar systems, it is also possible to reduce the effort required to tailor a system to satisfy specific needs.

Given a family of systems, a system which belongs to that family is sufficiently distinguished entirely by determining how the family's variabilities are resolved with respect to that system. By viewing the variabilities as unresolved requirements and engineering decisions, building a particular system can be reduced to resolving these decisions in such a way that the resulting system will satisfy particular customer requirements. This is analogous to the concept of rapid system prototyping in which a partial prototype already exists (in this case, corresponding to the aspects in which systems are similar).

Having identified the variabilities that characterize a family, it is feasible to represent the work products associated with the set of systems in the family so that each work product is tailorable to the corresponding requirements and engineering decisions and, therefore, mechanically derivable for any particular system. This capability has been used previously in the development of subsystems (chassis that support multiple body configurations and engines, or engines that can be used in multiple different vehicles), software (through the use of such techniques as macroprocessors, Ada generics, and metaprogramming), in user documentation (through the use of form-letter and metaprogramming mechanisms), and in computer hardware (through programmable read-only memory chips and plug-compatible device interfaces).

When mechanically deriving a system from selected requirements and engineering choices, it is much harder to understand the implications of all decisions than when the system is built entirely from

scratch. To remedy this limitation, there is a need to be able to analyze the implications of the mechanically derived system that results from these decisions. Model-based analyses support trade-offs among alternative mechanically derived systems that might all meet perceived customer needs but result in different performance, reliability, or usability properties. This capability:

- Requires an ability to model the target environment adequately
- Substantially reduces the uncertainties in fielding a complex system
- Allows extensive testing in the operational environment when testing in the real environment is not feasible
- Allows earlier experimentation and user validation before all systems, subsystems, and/or components are available

3. SYSTEMS ENGINEERING WITH GSEP

This section describes the GSEP. The purpose of this section is to provide the reader with a general understanding of the GSEP and how it meets its design objectives.

- Section 3.1 describes the GSEP at a very high level.
- Section 3.2 sets the context for the GSEP. This section describes what the GSEP addresses and describes the external influencing factors.
- Section 3.3 explains how the GSEP implements the fundamental concepts that are described in Section 2.
- Section 3.4 contains a description of the GSEP model's high-level activity flows.

The full GSEP description is given in Appendixes A, B, and C.

3.1 THE GSEP MODEL

The GSEP is a process model. It is used to create a process that can be made enactable through process tailoring. In this report, GSEP is used to refer to both the process and the process model. There are no differences between the GSEP model and the highest level process that is generated from it. Consequently, the statements made about the GSEP and the GSEP model would be identical and redundant. It is only through process tailoring of the high-level process that differences occur. See Section 4 for more information on process tailoring.

GSEP is a comprehensive systems engineering process model. There are four reasons that GSEP is considered comprehensive (see Section 2.2):

- ***GSEP is well defined.*** The GSEP activity definitions include a description of the activity, the entrance and exit criteria for the activity, and the inputs and outputs for the activity. The GSEP activity flow describes the activity ordering, the work product flow between activities, and the controls and mechanisms associated with each activity.
- ***GSEP is well managed.*** The GSEP contains a management subprocess that describes the management activities and addresses management issues.
- ***GSEP is scalable.*** The GSEP is a basic building block for developing systems. Part of the GSEP description is how GSEP is used to create processes for developing all parts of a system. Thus, GSEP can be used to construct a variety of systems from the simple to the complex.
- ***GSEP is a disciplined process.*** GSEP is disciplined in that it defines mechanisms for ensuring that the GSEP process is adequately followed.

The following discussion (Section 3.1.1) provides an understanding of the basic GSEP model and how it can be used to develop a system.

3.1.1 GSEP'S ROLE IN THE SYSTEM LIFE CYCLE

Figure 7 shows how GSEP fits into the phases of the system life cycle. (The system life cycle is described in Figure 1, Section 2.1.)

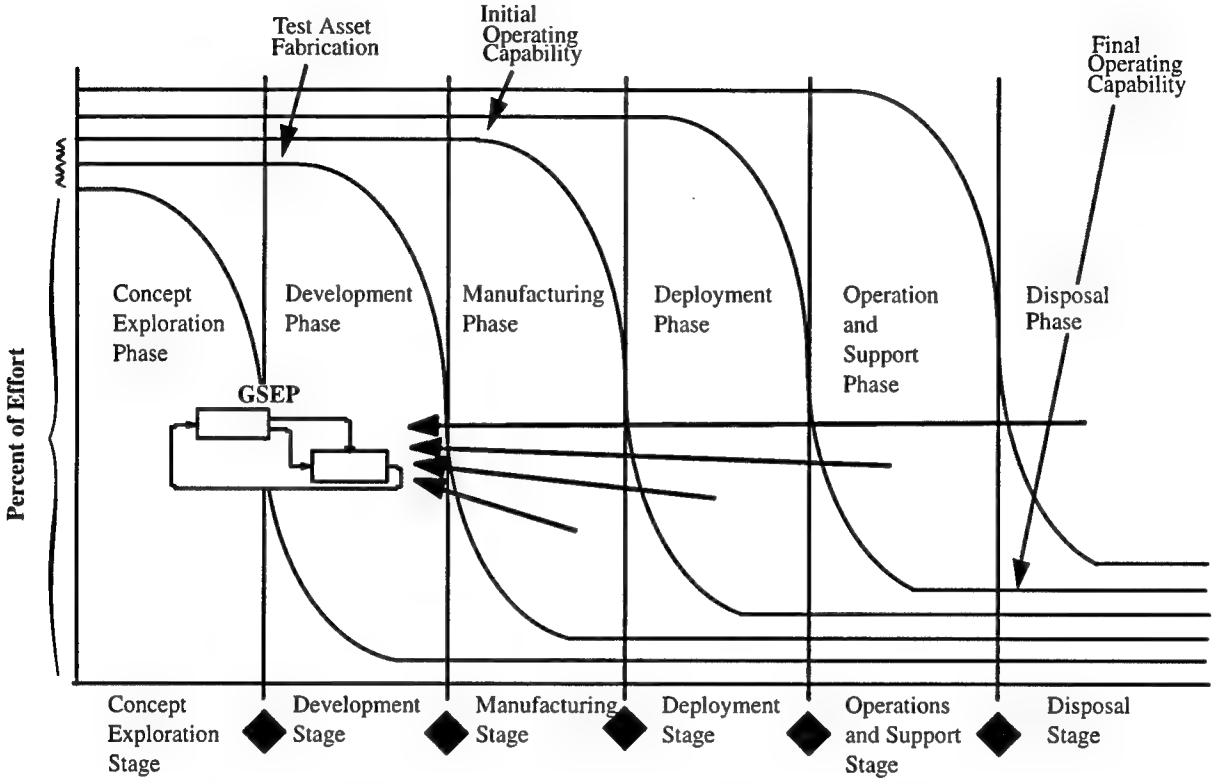


Figure 7. GSEP in the System Life Cycle

The GSEP is specifically designed to address the activities that are prominent in the concept exploration and development phases. These phases of the life cycle define the parts of the system to be developed (i.e., operational, manufacturing, support, etc.) and the processes for developing, implementing, and using each part. These phases do not include carrying out the processes for implementing and using the system parts. However, the GSEP development team believes that the GSEP is applicable to all system life-cycle phases, although further investigation is needed to support this belief.

3.1.2 A GENERIC DEVELOPMENT PROCESS MODEL

Section 2.2 describes development process models and defines a generic development process model. Because the GSEP is a generic development process model, it has the following two advantages:

- GSEP is applicable to a wide variety of systems development efforts.
- The GSEP is a basic building block and can be used in every stage to reach every product state.

Like other development process models, GSEP is defined by its activities and their order, the information flows and work products that are associated with the activities, and the GSEP architecture. The GSEP architecture defines how GSEPs are arranged to develop an entire system.

3.1.2.1 The GSEP Model Activity Set

Section 3.4.1 and Appendix B describe the management and technical activities.

3.1.2.2 The Information and Work Product Flows

In GSEP, the information and work product flows provide an understanding of what information and work products are used in activities as inputs, controls, mechanisms, or outputs. The following list describes each type of work product:

- ***Inputs.*** Information or work products that, if available, are used by the activity to perform its function. If a work product or piece of information is not available, then either the activity is performed without it, or the activity waits until the missing work product or information is available before beginning. The entry criteria determine which strategy is used.
- ***Outputs.*** Information or work products that may be produced or modified by the activity. The exit criteria determine whether creation or modification of a work product or piece of information is mandatory or optional.
- ***Controls.*** Controls are information or work products that are used to constrain or direct the work performed during the activity. For example, a management plan is a control for a technical activity because the technical activity is required to follow the schedule documented in the plan.
- ***Mechanisms.*** Mechanisms are used during an activity in order to accomplish the activity's function; for example, people, facilities, and tools are mechanisms.

GSEP also uses information and work product flows in conjunction with the activity entry and exit criteria to establish a partial ordering of the activities. This ordering is important because it determines the activity dependencies.

3.1.2.3 The GSEP Process Architecture

The GSEP process is used iteratively to carry out each development increment (e.g., build 1, build 2, build 3, etc.). A GSEP is instantiated for each system part (e.g., system, subsystem, component). How these GSEPs are arranged and the interfaces that are established between them are important part of architecting the process.

3.1.3 Tailoring GSEP

To apply GSEP to a particular systems engineering effort, it must be tailored. Section 4 discusses GSEP process tailoring. During process tailoring, the engineer establishes the system context and identifies process drivers (i.e., objectives and constraints on the process). The tailored GSEP process is then instantiated (i.e., specific resources, methods, tools are assigned), and when the instantiated process is carried out, it produces the system.

During tailoring, decisions are made that specify the exact order (further constraining the partial ordering) of the activities and the specific work-product content and information flows. However, the generic activities and flows defined in the GSEP are not modified.

3.1.4 Understanding IDEF0 Diagrams

Section 3.2 establishes the context for the GSEP model that is further defined in Section 3.4. To understand the GSEP model, it is important to understand the IDEF0 notation (U.S. Air Force 1981) that the diagrams follow. The following list explains IDEF0 notation and conventions:

- ***Notation.*** IDEF0 is a methodology that was used to create the GSEP activity flows. Its notation consists of boxes, lines with arrowheads (arrows), and text.
  - **Boxes** represent activities.
  - **Arrows** represent information or work-product flow between activities. The information flow is in the direction of the arrow.
  - **Text** is used for descriptive labels inside the boxes that indicate the activity names and numbers and the information or work-product name associated with each arrow.
- ***Decomposition.*** Activities are decomposed into a series of lower level diagrams depicting the subactivities that make up the higher level activity. In the lower right corner of each activity box is a letter-number combination that identifies the activity by number. In Figure 8, the x.x.x represents the activity number. The activity number is used to identify where the activity belongs in the decomposition hierarchy. For example, if the activity numbered 1.1.2 is decomposed into three other activities, they are numbered 1.1.2.1, 1.1.2.2, and 1.1.2.3, respectively.

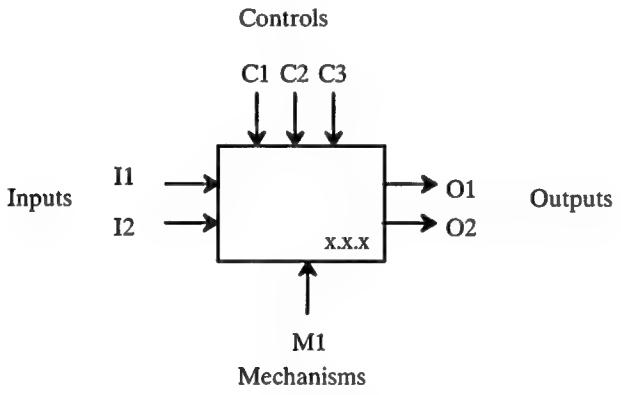


Figure 8. IDEF0 Arrow Definitions

- ***Information/Work Product Flows.*** In IDEF0 notation, arrows can enter an activity from any of four directions: left, top, right, or bottom. Arrows that enter from the left represent activity inputs, arrows that enter from the top represent activity controls, arrows that exit to the right represent activity outputs, and arrows that enter from the bottom represent activity mechanisms (see Figure 8). These four types of information flows and work products are explained in Section 3.1.2.2.

  When information and work product flows are connected to other diagrams, IDEF0 uses an identification convention to reduce ambiguity. This identification uses the first letter of the information or work product flow type followed by a number. The number is consecutive starting with the top or right side of the diagram. For example, in Figure 8 the Inputs are labeled I1 and I2, the outputs are labeled O1 and O2, and the controls and mechanisms are labeled C1, C2, C3, and M1, respectively.

- ***Diagram Header Information.*** When activities are decomposed, the parent activity's number (i.e., Node), name (i.e., Title), and revision number are in the activity header at the top of the IDEF0 diagram (see Figure 9).

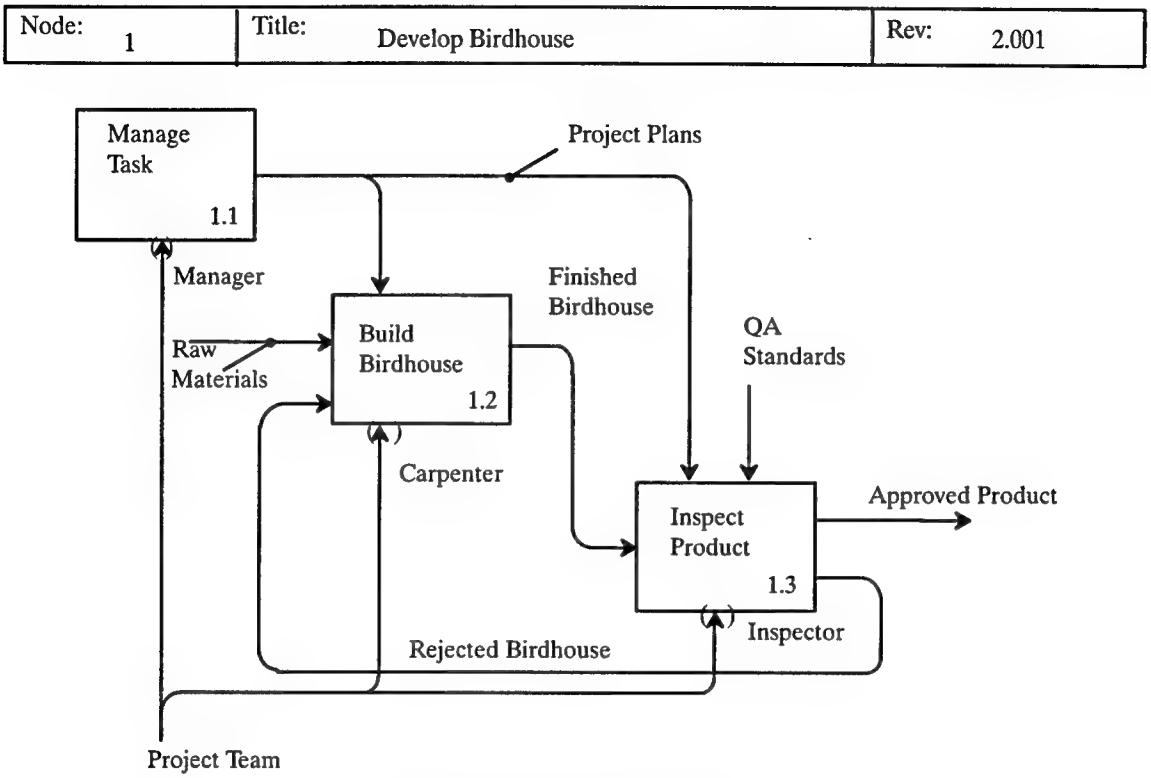


Figure 9. Sample IDEF0 Diagram

- ***Tunneled Information.*** Arrows with parentheses ( ) indicate "tunneled" information. Tunneling is a type of shorthand notation. If an information or work product flow is tunneled into or out of an activity, it implies that all decompositions of the activity also have the same information or work product flow, even though it is not shown explicitly on their diagrams. For example, in the IDEF0 example in Figure 9, the Manager is a tunneled mechanism into the Manage Task (1.1) activity. If the Manage Task activity is decomposed into five other activities (not shown on this diagram): Define Birdhouse Context (1.1.1), Analyze Construction Risks (1.1.2), Plan Birdhouse Construction (1.1.3), Monitor Birdhouse Construction (1.1.4), and Assess Birdhouse Construction Progress (1.1.5), then each of these five activities has the Manager as an implied mechanism. The decomposed IDEF0 diagram (Manage Task, 1.1) does not explicitly show the Manager as a mechanism. Tunneling is used to keep the diagrams from being cluttered with redundant arrows.

The GSEP Enterprise Context diagram (see Figure 10) outlines some additional conventions that are used in the IDEF0 GSEP diagrams.

3.2 SETTING THE CONTEXT FOR THE GSEP

Figure 10 contains the decomposition of the Enterprise Context activity (−0).This activity defines the external context for the GSEP. Its purpose is to provide a frame of reference for the highest level GSEP activity, Develop Systems (0). The Manage Organization activity, although very important to the enterprise, is not part of the GSEP, and is not decomposed in this report.

The inputs and controls that impact the Manage Organization activity are shown only to set context and not to imply that these are the only inputs or controls that are required to manage the organization (see Figure 10). The inputs, outputs, controls, and mechanisms to the Develop Systems (0) activity are generic. This enables the highest level GSEP activity to be adequately tailored for use in a variety of systems development applications. For example, in the Automobile System (Section 2, Figure 5) the Customer Requirements are the automobile dealer and/or the automobile purchaser requirements. In the Powertrain Subsystem, the Customer Requirements are the Car Subsystem requirements that have been passed down from above. All of the mechanisms on the context diagram (Figure 10) are tunneled. This is because the resources, methods/algorithms, and tools are mechanisms for all of the GSEP activities, and tunneling them in this context diagram implies their existence in all GSEP activity diagrams.

The Develop Systems activity (0) can be interpreted as the organizational endeavor to develop multiple systems, related or unrelated. Related systems may be technical spin-offs or similar systems that exist in a product family.

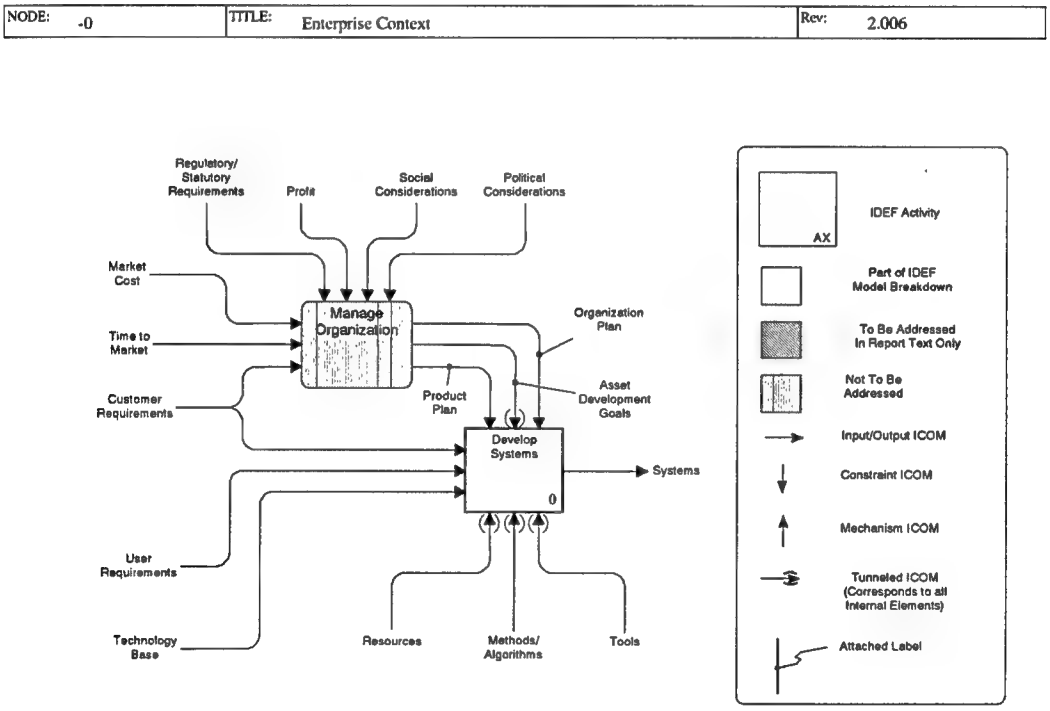


Figure 10. Enterprise Context Diagram

Figure 11 contains the decomposition of the Develop Systems activity (0). This activity identifies the customer requirements, user requirements, and technology base used to develop the system. The high-level product and organizational plans (i.e., External System Definition, Subsystem/Component Status, and External System Plan) impact the way in which the system is developed. If the inputs and controls come from the level above (i.e., Above), they constrain the Develop System activity (1). If the inputs and controls come from the level below (i.e., Below), they inform and may result in the need for appropriate action in the Develop System activity (1). If the inputs and controls come from across (i.e., Across), they either constrain or inform, as appropriate. The Develop System activity is responsible for developing all the systems that the organization will produce.

This activity also defines the interactions between higher level systems, subsystems, and components (see Section 2.3.2). Subsystems will receive information from the system or subsystem above and will send data to components or subsystems below. This activity also describes how information is communicated between peer activities using "across" information flows. Components will only receive data from systems or subsystems above them or subsystems/components across from them.

The interaction of subsystems or components with their peer subsystems or components is important for efficient communication of information and coordination of activities. This type of interaction is used to mitigate interface design issues and technical inconsistencies or problems in the system design. The GSEP supports this type of interaction. There are times when very tight interactions between somewhat disconnected subsystems are required; for example, when an operational component must be manufactured by a manufacturing component. The engineering team defining the operational system has to ensure that it can be manufactured, and visa versa. These interactions must be carefully monitored to ensure that system design changes do not adversely impact the design decisions that were made during the selection of the overall system design.

These interactions could be controlled by forcing all of the intersubsystem communications to formally go through the system above, but this would likely stifle the communication and beneficial creativity. On the other hand, it may be difficult for engineers with a subsystem- or component-level perspective to discern when a discussed/proposed change adversely affects the system design above. As a result, the engineers may not feel that the change warrants exposure to the upper level system, and the problems may not be found until later system integration.

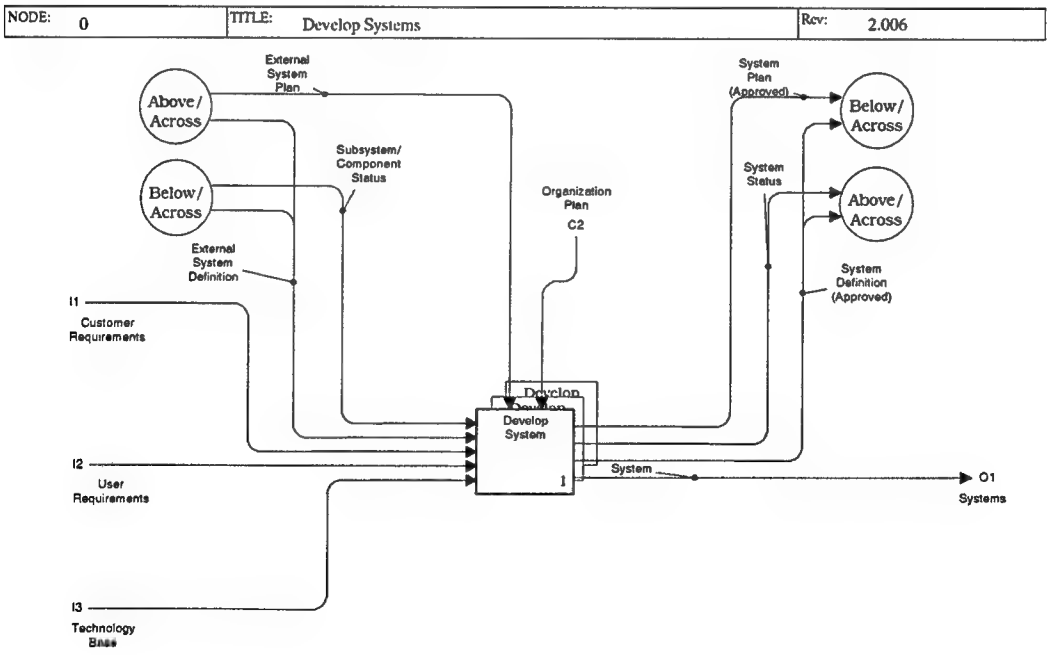


Figure 11. Develop Systems

Several systems may be developed during the Develop Systems activity (0); these are represented by the shadow boxes behind the Develop System activity (1) in Figure 11. The inputs, outputs, controls, and mechanisms for the other Develop System activities (1) (i.e., the shadow boxes) are identical. If integration between the Develop System activities (1) is necessary, it would be established in the same way that across information flow is achieved between subsystems. In the description of the GSEP model contained in the remainder of this section, it is assumed that there is one Develop System activity. This assumption is not limiting, because all Develop System activities are based on the same GSEP definition.

The Develop System activity (1) includes the management, technical, and implementation subprocesses. For the purposes of the GSEP, Develop System is directly related to the "Development" phase of a system's life cycle. GSEP includes the decision processes required for optimizing and fully describing a system so that it can be fabricated and used in the manner in which it was intended. The GSEP does not include the development activities used to implement the system definition, but does include the interfaces to the implementation activities.

Figure 12 contains the decomposition of the Develop System activity (1). This activity manages the development effort, defines the system, and implements the system that is to be built. The GSEP portion of this activity is shaded and includes activities 1.1 and 1.2. This portion includes the management of the system development and the technical activities that produce the system.

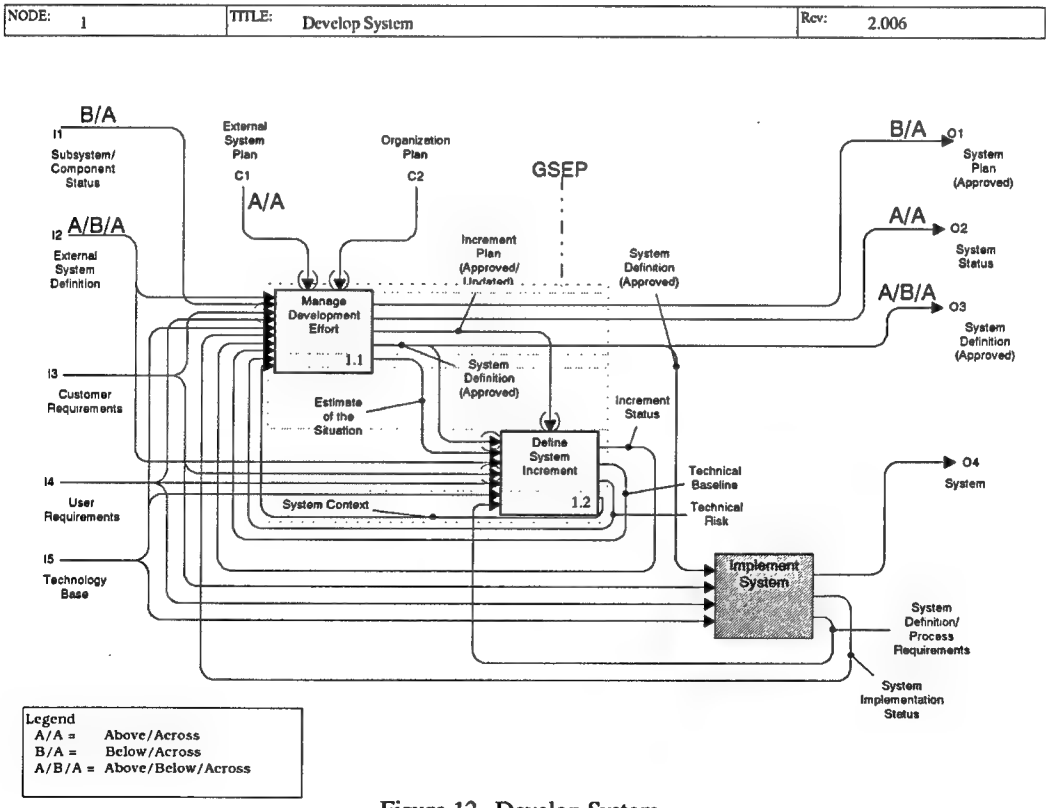


Figure 12. Develop System

3.3 ADDRESSING SYSTEMS ENGINEERING CONCEPTS WITH GSEP

Fundamental system engineering process concepts were described in Section 2.3. Part of understanding GSEP is understanding how the GSEP addresses each of these concepts. Section 3.3

describes the GSEP approach to integrating management and technical activities, dealing with system complexity, supporting concurrent engineering, separation of concerns, and systems architecture.

3.3.1 Integration of Management and Technical Activities

GSEP addresses the integration of the management and technical activities (see Section 2.3.1) in the systems engineering process. The GSEP defines a set of generic management activities as well as a set of generic technical activities. GSEP specifies two types of integration of the management and technical activities: within a system part (e.g., system, subsystem, component) and between different system parts (see Figure 13). The Manage box refers to the management activities, and the Technical box refers to the technical activities. The lines connecting the Manage and Technical boxes represent the management and technical integration within system parts. The lines connecting the different system parts (e.g., connecting the System-Level GSEP and the Subsystem-Level GSEPs) represent the integration of the management and technical activities that occurs between different system parts.

Integration between system parts includes the above and below integration (e.g., subsystems interfacing with the system or subsystem above and/or subsystem or component below) and the across integration (e.g., integration between peers). The lines for the across integration are not shown in Figure 13, but do exist. The above and below integration is established by the higher level system part (e.g., the system defines the integration between the system and its immediate subsystem constituents). The across integration is established at the highest level where the need for that integration is visible.

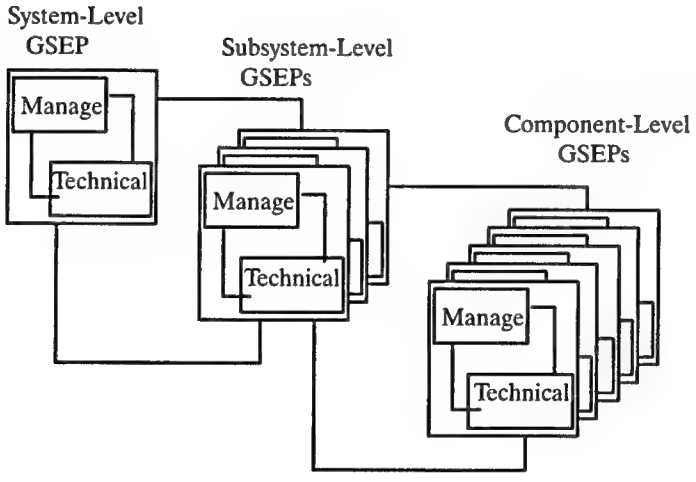


Figure 13. GSEP Management and Technical Activity Integration

Figure 14 describes the integration of the management and technical activities within a system part. The approved System Definition and Estimate of the Situation (EoS) are input from the management activities into the technical activities. The approved Increment Plan is a control into the technical activities. The Technical Risks, Technical Baseline, Increment Status, and System Context are output from the technical activities and input into the management activities.

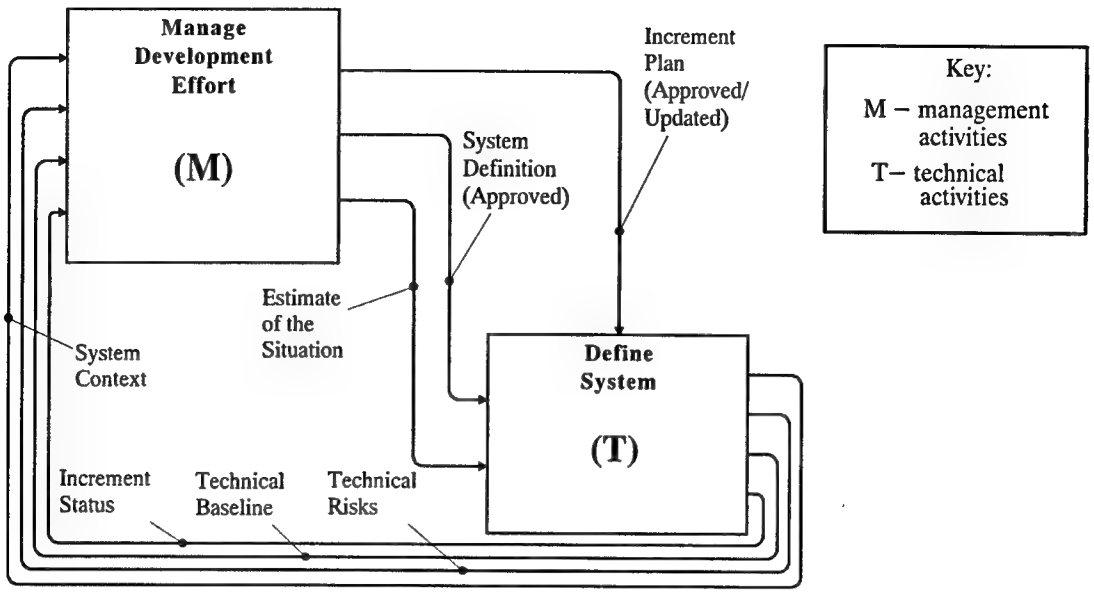


Figure 14. Management and Technical Integration

3.3.2 Dealing With System Complexity

The GSEP deals with the complexity and scale-up (see Section 2.3.2) of a systems engineering effort through **system decomposition.** GSEP addresses managing the technical activities through the development of **system increments**. System decomposition is supported in the GSEP by defining how a system process interacts with higher or lower level "system processes" (see Figure 4, Section 2). The definition of a system is based on perspective. To the engineer who must manage and/or create the system, its constituent parts are the subsystems, components, or elements; but to the engineer who must manage and/or create the subsystem, that subsystem is the system. The GSEP addresses system increments by separating management concerns that address an increment of system development from management concerns that address the sum of all increments that compose the system development.

3.3.2.1 System Decomposition

The GSEP concretely defines the process interfaces between the levels in the system decomposition. Figure 15 identifies the information that is passed between system, subsystem, and component processes. "Above" identifies communication with other systems that view this system as a subsystem or component. "Below" identifies communication between a system and its constituent subsystems and/or components. These communications include the following:

- ***External System Plan.*** This plan comes from above or across. If it comes from above, it constrains the plan for this system; if it comes from across, it informs or constrains the plan for this system.

- ***Subsystem/Component Status.*** This input contains the current status of a subsystem or component. If it comes from below, it is the status for one of this system's constituent parts; if it comes from across, it is the status for one of this system's peers.

- ***External System Definition.*** This input is the System Definition. If it comes from above, it is the system definition of the system above this one; if it comes from below, it is the system definition of the system below this one; and, if it comes from across, it is the system definition of a peer system.

- ***System Plan.*** This output is the System Plan that will constrain other system parts. If it is output below, it constrains this system's constituent parts; if it comes from across, it constrains other peer systems.

- ***System Status.*** This output is the status of this system, and it is reported to the customer or a system process above or across from this system. If it reports to a system process above, it is used to inform that system of this system's current status; if it reports across, it informs other peer systems of this system's status.

- ***System Definition.*** This output is the work product that is developed during the Develop System activity (0). The System Definition is the primary output of the GSEP. If it comes from above, it is incorporated into the next higher level system; if it comes from below, it constrains the system definition of this system's constituent parts; and, if it comes from across, it informs other peer systems of this system's definition.

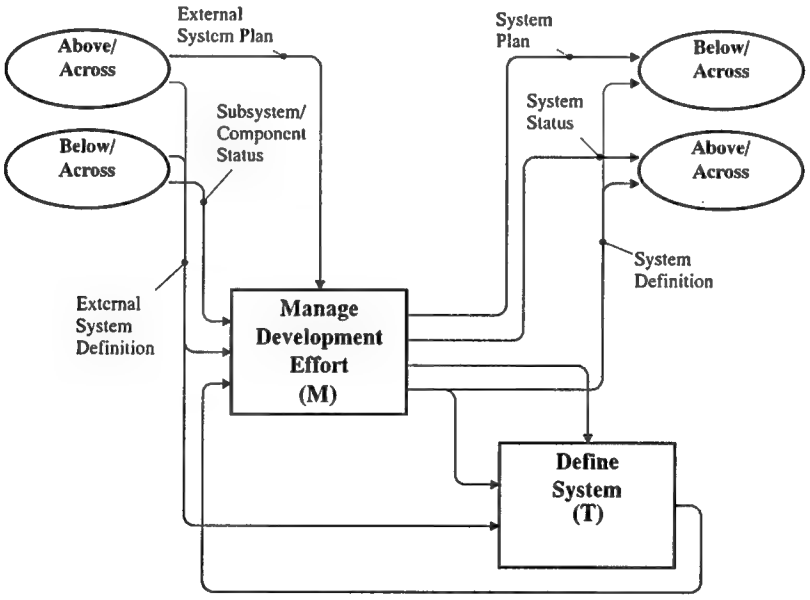


Figure 15. GSEP Approach to Dealing With Complexity

Figure 15 also contains unlabeled arrows that show the integration between the GSEP management and technical activities. These unlabeled arrows are representative of the information and work product flows identified in Figure 14.

Figure 16 contains an example of how the GSEP can be used to establish information and work product flows between different levels of system decomposition. In this example, the system management has oversight of the subsystem, and the subsystem has management oversight of the component. The subsystem in the example represents the most complete mapping of GSEP inputs and outputs between system levels.

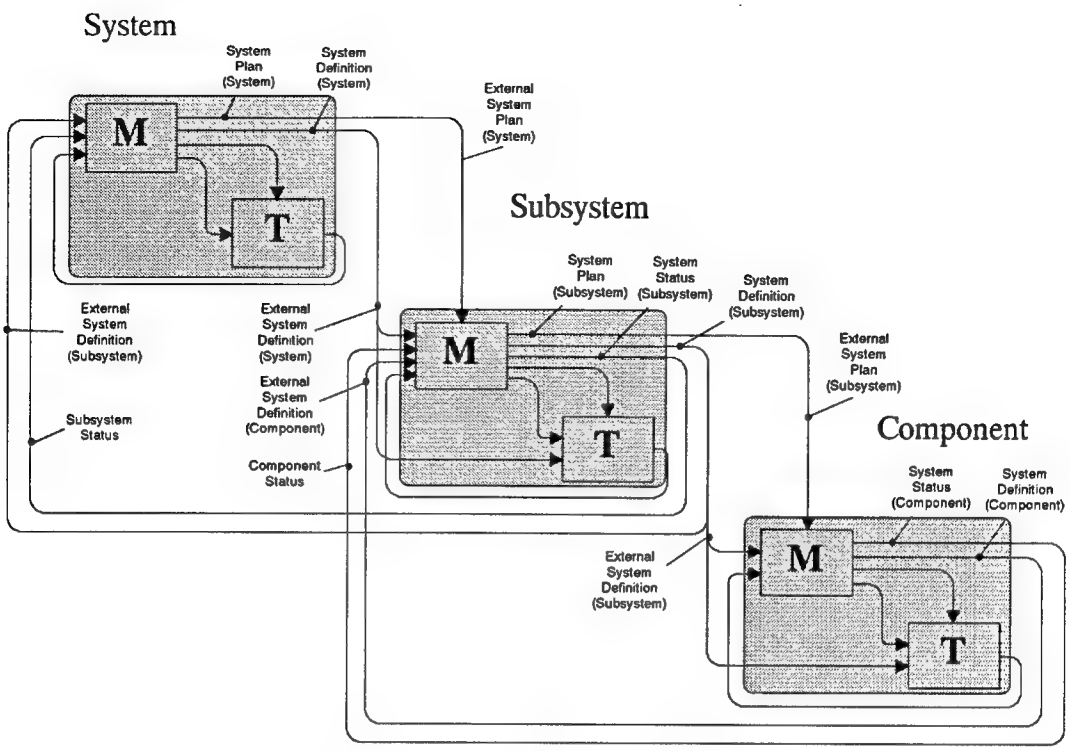


Figure 16. Interactions Between Levels of Decomposition

One other aspect of complexity that the GSEP addresses is the across communication between peer system parts. Figure 17 is an example of this type of interface for the Automobile System (See Figure 5). Although Figure 17 is an example of how communications with other system parts can be defined for the Automobile System, it is representative of the interactions between any two instances of the GSEP.

3.3.2.2 System Increments

The GSEP identifies management activities that are concerned with the system increment and other activities that are concerned with the sum of all increments. When defining the system development increments, it is important to consider how to best divide the development objectives for the system (e.g., Powertrain Subsystem) into reasonable pieces. Thus, defining the increments results in determining which of the GSEP technical activities will be completed in each increment. Every increment must contain all of the GSEP management activities including the assessment of the increment's impact on the completion of all other system increments.

Using the Powertrain Subsystem as an example, the objectives for developing the first increment of the powertrain might be to complete the requirements definition, including the technical activities for analyzing the powertrain needs, and to define the powertrain requirements in addition to all of the GSEP management activities. If the objective of the first increment is to define a conceptual prototype of the powertrain, the first increment includes all of the GSEP technical activities in addition to all of the GSEP management activities.

The increments for the subsystems and components must be compatible with the system increments. Peer subsystem/components may also need to be compatible to ensure that incremental system development objectives can be met.

In Figure 17, the drive Shaft Component that is defined in the operational subsystem needs to be manufactured. This has both system decomposition and system increment implications. The system decomposition must ensure that the system definition for the drive shaft is considered in the definition of the manufacturing subsystem that is responsible for creating the drive shaft. The drive shaft component must also take into consideration design decisions that may prevent the drive shaft from being manufacturable. As a result, information must be exchanged between the Drive Shaft Component and the manufacturing subsystem to ensure the successful production of the automobile.

The Automobile System management activities are responsible to schedule the development of the Drive Shaft and the Drive Shaft Components in a reasonable fashion. The system increment must ensure that if the first Automobile System increment develops the drive shaft, then the Manufacturing Subsystem must support the drive shaft development in that increment. Note that if the manufacturing of the drive shaft is high risk, a separate GSEP may be instantiated specifically to deal with the integrated set of requirements for the high risk portions of the drive shaft and its manufacture. (This decision is made by the Automobile System.) The system definition that results from this integrated effort becomes input into the GSEPs for the Drive Shaft Component and the Manufacturing Subsystem as an external system definition.

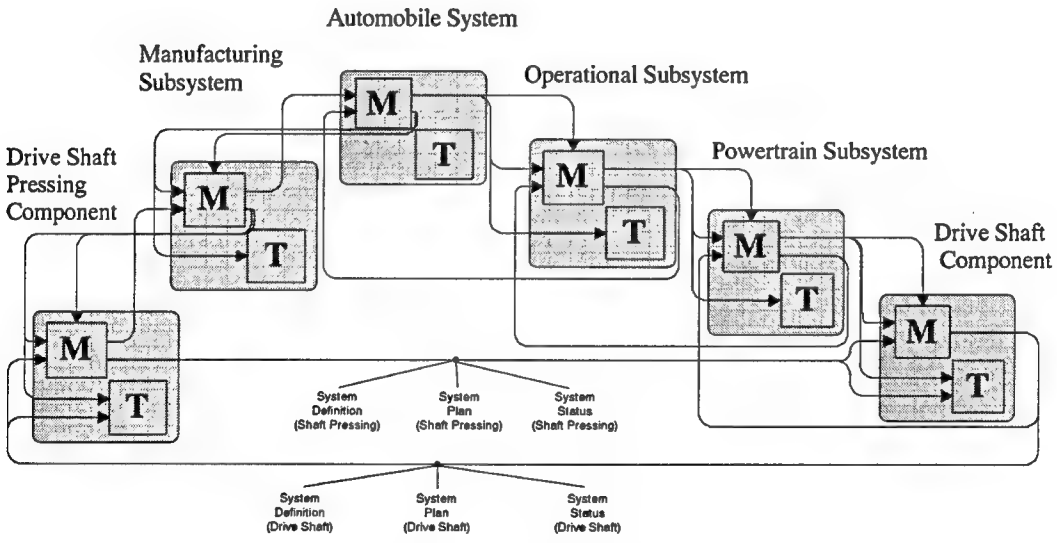


Figure 17. Across Communications Between Systems Engineering Processes

3.3.3 Support for Concurrent Engineering

The GSEP supports all three types of IPTs identified in Section 2.3.3. The integration of the disciplines within the GSEP is supported by the work products and activities within the GSEP. The product teams are defined by people allocated as resources to multiple activities in the GSEP. The teams focus on common objectives and share knowledge and expertise as appropriate. Process interactions between the subsystems/components define the Type 2 and 3 IPTs. A large amount of interaction between the

processes that create the subsystems/components implies a need for integration of resources. This integration can be achieved through IPTs. The key integrating activities include the following:

- **Management and technical overlap** is when technical resources that understand the technical needs and concerns work with management resources that understand the management issues and the system context. An example of this type of grouping is when the technical resources involved in identifying technical risk work with the management resources that analyze the system risk.

- **Overlap within management activities** is when management at different system levels (e.g., increment planning and system planning) work together. An example of this type of grouping is when management resources that defined the external system plan work with those that perform the risk analysis.

- **Overlap within technical activities** is when technical resources at different system levels, or within a system level but in a different discipline, work together (e.g., resources responsible for the development of the alternative architectures and those resources responsible for evaluating those architectures [ilities]). An example of this type of grouping is when resources responsible for analyzing needs, analyzing requirements, and validating/verifying the solution work together.

3.3.4 Separation of Concerns

The GSEP addresses the separation of concerns that are discussed in Section 2.3.4 through its ability for base management and technical decisions on the defined objectives for a work product. For example, as stated in Section 2.3.4, the statement of needs is often used to create a set of high-level functions (i.e., informal functional hierarchy). The objective of these functions is to generate the initial set of requirements which provide the basis for additional decomposition to reach a set of detailed requirements. Although these detailed requirements could be further decomposed until a design emerges, this is not their objective. The design that emerges using this approach may not meet other objectives because an assessment of the design objectives was not performed.

In GSEP, the objectives of work products are clearly stated, and work products are only used in ways that are appropriate based on their creation objectives. Therefore, concerns are always kept separate and the work performed is always toward a set of objectives identified specifically for that effort.

The IDEF diagram in Figure 18 illustrates how the GSEP addresses this separation of concerns. The Develop Informal Functionality activity takes the problems and needs and identifies an Informal Functional Hierarchy that is then used to derive and flush out the system requirements. The Informal Functional Hierarchy is usually very dependent on historical/existing systems that have been built. As a result, the functions are often unnecessarily constrained by the historical systems.

The requirements are then developed using these functions. The Define Requirements activity drives these functions down into a testable set of system-level requirements that are traceable to the informal functions. The definition of the functional architecture then proceeds using the identified requirements rather than starting with the informal functions. The partitioning of requirements into functions will then define functions that are based on what this system has to do rather than the historical functions defined in the informal functional hierarchy.

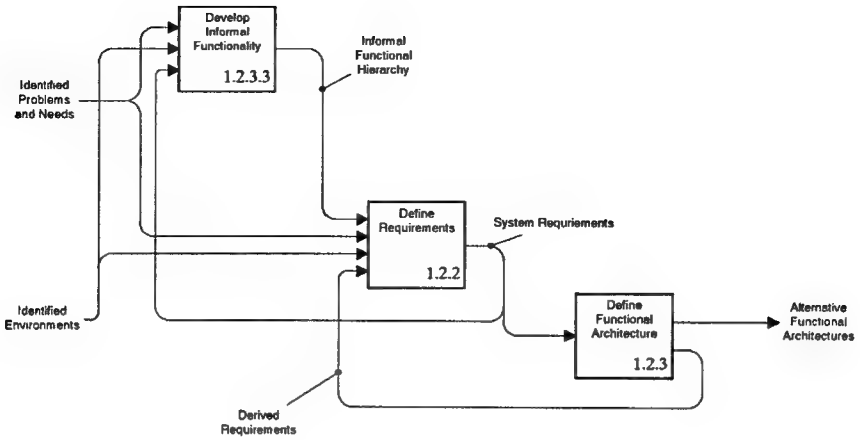


Figure 18. Separation of Concerns

3.3.5 Systems Architecture

The GSEP partially addresses the issue of systems architectures (see Section 2.3.5). In the GSEP, the Define Functional Architecture activity (1.2.3) (see Figure 28, page 56) defines the set of subactivities that create the allocation-independent functional architecture. In these activities, a broad set of alternative functional architectures that adequately explore the design space are defined. The GSEP does not address how these architectures are created. Methods chosen to carry out these activities should address the creation, derivation, or discovery of these architectures.

Evaluation of the effectiveness of these functional architectures is directly dependent on the allocated architectures that are derived from them. As a result, care should be taken to select alternative allocated architectures that are viewed as representatives of the alternative functional architectures.

3.4 GSEP DESCRIPTION

This section contains a description of GSEP. It also contains a description of all the GSEP IDEF0 diagrams in Appendix A, with the exception of the context-setting diagrams described in Section 3.2. Section 3.1.4 contains a description of the IDEF0 notation. Detailed activity descriptions and work product descriptions can be found in Appendixes B and C, respectively.

Figure 12 contains the decomposition of the Develop System activity (1). This activity manages the system development effort, defines the system, and implements the system definition. The system may be the manufacturing, support, operational, etc. "subsystems." This activity includes both the management of the system development and the technical activities that produce the system.

The Implement System activity (1.3) produces the actual system by carrying out the System Definition that is defined in the GSEP process. No aspect of the system definition is to be modified in this activity. Instead, there must be iteration back into the GSEP activities (i.e., Manage Development Effort [1.1] and Define System Increment [1.2]) to make modifications to the System Definition. The activities that implement the system definition are not included in GSEP.

The Manage Development Effort activity (1.1) determines the context of the system development, analyzes risks in creating the system, develops an increment plan that defines a plan to reach a milestone in the development of the system definition, tracks the technical development of the system, and develops a system plan that manages the development of system definition for this system and any subsystems or components for which this system is responsible.

The Define System Increment activity (1.2) is the technical subactivity of the GSEP and is responsible for transforming a set of needs into a complete, balanced, system solution. All the GSEP technical activities are subactivities to this activity. This activity determines user and customer needs, defines systems requirements, defines a functional architecture, synthesizes the allocated architecture, selects a life-cycle optimized solution, and validates and verifies the system solution. This activity produces a verified and validated system design that is used to specify subsystems or components, or it may be directly implemented. When subsystems and/or component definitions are completed, the "as built" definitions and evaluation results are passed back up to the Develop System activity (1) and used to update and evaluate the system design.

The next two sections, Sections 3.4.1 and 3.4.2, describe the GSEP management and technical activities, respectively.

3.4.1 Manage Development Effort

Figure 19 contains the decomposition of the Manage Development Effort activity (1.1). This activity defines the management subprocess of the GSEP and includes activities related to the analysis of program risk, process tailoring, planning and scheduling, tracking the technical subprocess, and controlling and managing the entire systems engineering effort, potentially including the development of some or all of the constituent subsystems and components.

The Manage Development Effort activity is responsible for providing the delivery of the developed system. The system definition is always delivered through management rather than by the technical activities.

The Understand Context activity (1.1.1) identifies and reviews all relevant information that could have an influence in the definition of the system. This activity is responsible for establishing what is to be pursued for the current development increment. This activity includes identifying the objectives, constraints, and alternatives that will be used in follow-on activities. Understanding Context includes defining the approach, estimating the situation, and reviewing the context definition.

The Analyze Risk activity (1.1.2) identifies potential long- and short-term risks. The process drivers identified in the EoS should be analyzed along with the risks to evaluate their impact on the increment's development process. Mitigation strategies are developed for each significant risk. To ensure that all of the significant risks are identified, lessons learned from previous, similar programs are reviewed, and a risk questionnaire, if available, is utilized. All increment stakeholders should have input into this activity.

The Plan Increment Development activity (1.1.3) uses the tailoring strategy and Risk Management Plan (RMP) to produce an increment plan. The increment plan defines the detailed plan for developing the increment and monitoring and reviewing the development. This activity reviews the significant increment risks and associated risk aversion activities and uses this information to create the increment process definition, plan, and schedule.

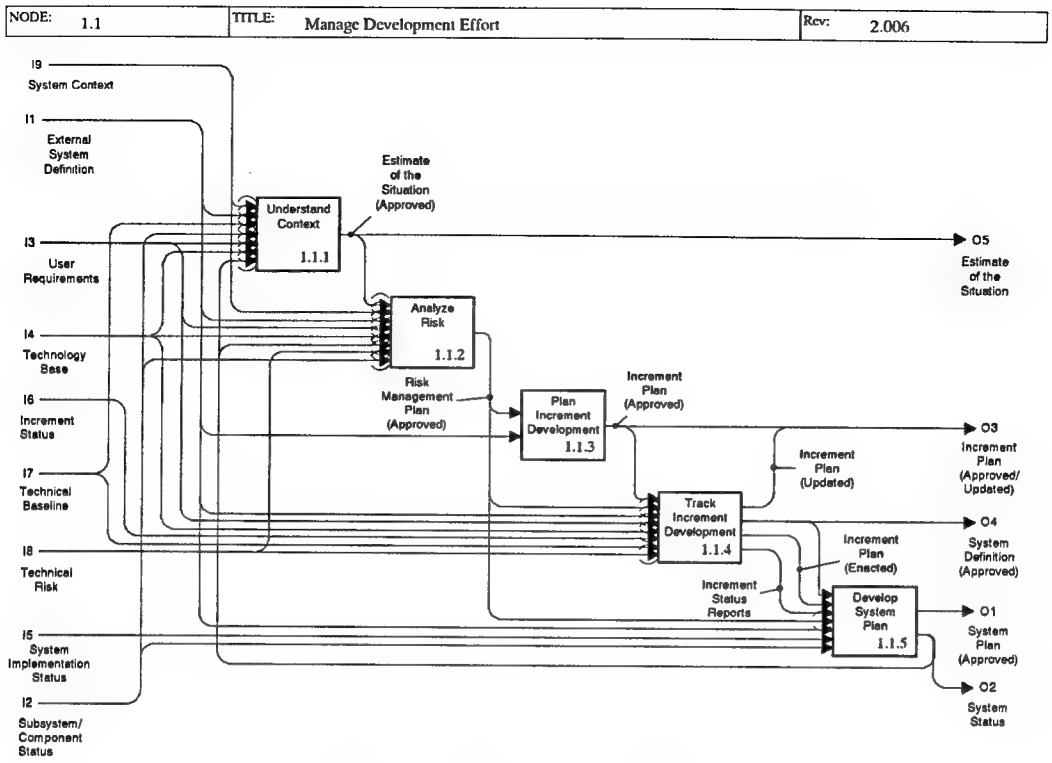


Figure 19. Manage Development Effort

The Track Increment Development activity (1.1.4) is the main management interface into the technical activities. This activity uses metrics provided by the technical systems engineering activities to assess progress of the technical activities. Minor modifications to the increment plan created in the Plan Increment Development activity (1.1.3) are permitted, but if major replanning is necessary, this activity is terminated and the Develop System Plan activity (1.1.5) is initiated. This activity includes a review of the technical product being produced during this increment. This review includes a comparison of the completed work products with the increment's success criteria. All officially delivered system definitions are distributed through this activity.

The Develop System Plan activity (1.1.5) defines the long-term plan for the development of foreseeable system increments, including the planning constraints for constituent subsystems and components. This activity coordinates the planning and statusing of the system as well as its subsystems and components.

3.4.1.1 Understand Context

Figure 20 contains the decomposition of the Understand Context activity (1.1.1). This activity defines the context of the system being developed including all of the external constraints, the EoS, and review of the context definition.

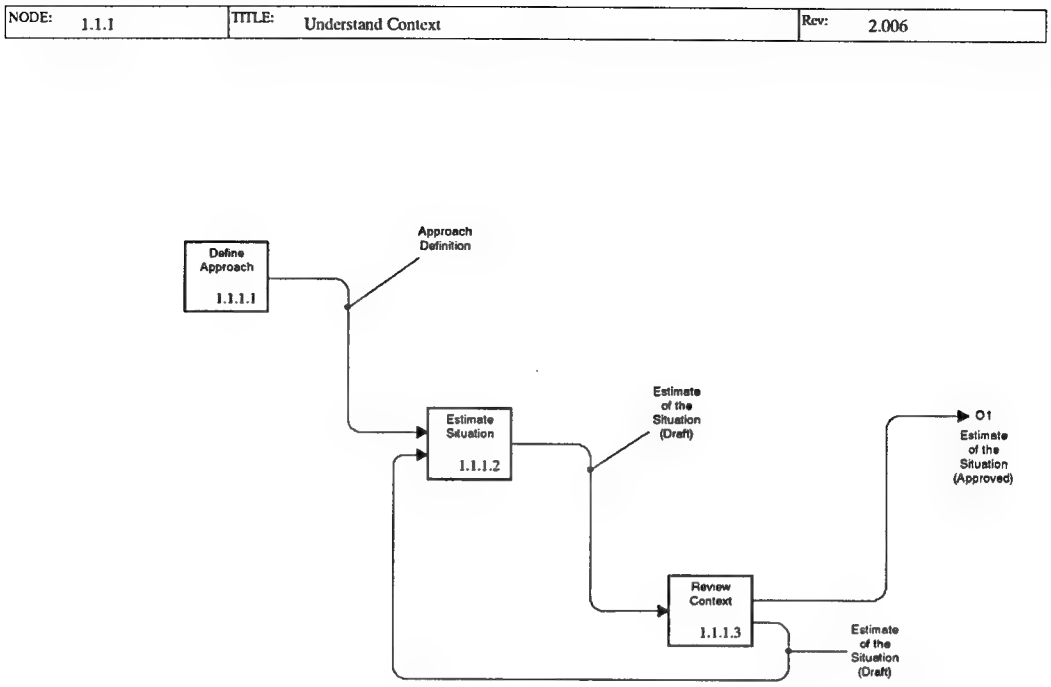


Figure 20. Understand Context

The Define Approach activity (1.1.1.1) defines key system/increment objectives, identifies system/increment constraints and stakeholders, and develops alternatives for meeting the system/increment objectives. This information is used by the Estimate Situation activity (1.1.1.2). The approach defined is based on the system context and the available resources.

The Estimate Situation activity (1.1.1.2) defines the EoS based on the approach definition generated by the Define Approach activity (1.1.1.1). The EoS documents assumptions, decisions and their rationale, and the approach definition.

The Review Context activity (1.1.1.3) takes the EoS and validates it against the stakeholder's expectations. This activity is also where the stakeholders commit to the system objectives and the approaches for meeting the objectives. This activity approves the EoS that is used in the risk assessment, the increment planning, and the development of the system plan.

3.4.1.1.1 Define Approach

The Define Approach activity (1.1.1.1) defines the objectives and identifies the alternatives, constraint, and stakeholders. Objectives must be results-oriented, clear and concise, controllable, measurable, reasonable, and appropriate for the current level of understanding about the system. Alternatives are different ways to meet the objectives. Constraints are limitations on alternatives. Stakeholders are individuals with a vested interest in the system being developed (e.g., customer, senior management, end-user). The objectives, alternatives, constraints, and stakeholders are used to establish a context for the system/increment.

3.4.1.1.2 Estimate Situation

The Estimate Situation activity (1.1.1.2) creates or updates the EoS. At a minimum, the EoS should identify and document the system approach defined in the Define Approach activity (1.1.1.1). The EoS may also document assumptions, decisions and their rationale, important historical information, and factors that may inhibit the successful development of the system. This information can be identified from sources like the contract and Statement of Work (SOW); insights about and from the client; historical project plans and budgets; client and business area policies, procedures, and standards; and interviews with key stakeholders at all levels.

3.4.1.1.3 Review Context

The Review Context activity (1.1.1.3) obtains stakeholder consensus on the defined approach including what is to be developed this increment. Specifically, this activity ensures that:

- All stakeholders are identified and their expectations are clearly documented.
- Objectives are clearly documented and are reasonable.
- Current increment-level objectives are derived from or are refinements of any higher level system objectives.
- Proposed alternatives adequately address objectives and constraints.
- All stakeholders agree to pursue the system and current increment.

All key stakeholders review and approve the EoS. Although the EoS does not need to circulate widely outside the immediate development team, senior management should review, correct, and approve the EoS to ensure that the documented objectives are clear and correct.

The EoS may be modified as a result of reaching agreement. All changes are documented along with the rationale for making the change, and the final, approved EoS is distributed to all stakeholders.

3.4.1.2 Analyze Risk

Figure 21 contains the decomposition of the Analyze Risk activity (1.1.2). In this activity, risks are analyzed, risk aversion strategies are developed, and stakeholder commitment is made on aversion strategy execution.

The Perform Risk Analysis activity (1.1.2.1) determines the likelihood and impact of the risks associated with the current increment of the system development. These risks are used to refine the process drivers as well as plan the risk aversion strategies.

The Review Risk Analysis activity (1.1.2.2) is where the results of the perform risk analysis are assessed. At this point, the goal is to have the team look for any unaccounted risks or risks that are infeasible.

The Plan Risk Aversion activity (1.1.2.3) determines strategies for reducing the impact of risks in the system and increment development. The RMP is updated to include risk mitigation plans and the approach that will be used to monitor risks during the system and/or increment development.

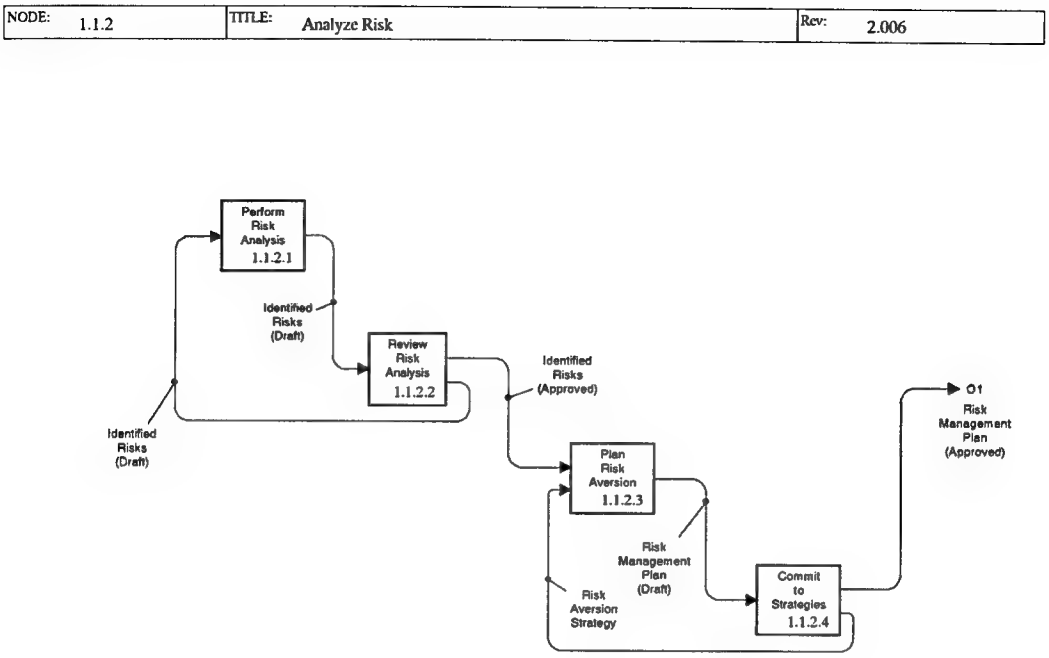


Figure 21. Analyze Risk

The Commit to Strategies activity (1.1.2.4) will validate the RMP. If the RMP is found lacking, it may be updated. This activity will also get the stakeholders to buy in to the RMP and tailoring strategies.

3.4.1.2.1 Perform Risk Analysis

The Perform Risk Analysis activity (1.1.2.1) is initiated by comprehensively identifying potential system and/or increment development risks. Objectives should be examined with respect to alternatives, constraints, and organizational and project assets, and potential risk areas should be identified. Unsatisfactory outcomes and the effect on both the current increment and system success criteria should be examined; that is, if a significant risk is identified for the current increment, it should be traced back to the system-level objectives and success criteria to determine whether the risk may affect the system as well as the increment. To help identify typical system risks, risk taxonomies such as in Boehm and Ross (1989, 117), U.S. Air Force (1988), and Charette (1990, 216–59) can be used.

Risks can be analyzed once they are identified by categorizing them and determining risk likelihood and consequence:

- Estimate the chance of potential loss (or gain) and the consequence (or benefit) of the risk situations previously identified.
- Analyze the risk dependencies (i.e., how one risk is impacted by another risk).
- Show any uncertainties in the risk likelihood and consequence estimates.

After completion of the risk analysis, a risk evaluation is performed to identify risk aversion strategies

and examine the impact of the risk aversion strategy on each high risk item. Any risk aversion strategy identified should reduce the cost and/or probability of risk occurrence to an acceptable level.

It is possible for risk aversion strategies to introduce new risks that detract from the anticipated benefits or negatively affect other risks; therefore, it is important to assess the impact of a risk aversion strategy on other risks.

In general, consider the following when defining risk aversion strategies:

- Is the strategy feasible?
- Are resources critical to technical success identified?
- Does the strategy reduce risk to an acceptable level?
- Will the strategy negatively affect another risk?
- What is the potential impact of new risks, if any, introduced by the strategy?
- Does the strategy support increment and/or system development objectives?
- Are the tactics and means for implementing the strategy consistent with increment and/or system constraints?
- Is the strategy cost-effective?

The results of this activity should be documented in the draft RMP. The plan is enhanced and matured during the remainder of the Analyze Risk activities.

Using New Technologies: A Technology Transfer Guidebook (Software Productivity Consortium 1993c) can help the engineer identify and evaluate strategies that involve new technology. This guidebook provides detailed guidance on identifying and selecting new technologies and defines the process activities necessary to insert the technology for maximum use and benefit.

3.4.1.2.2 Review Risk Analysis

The Review Risk Analysis activity (1.1.2.2) is an opportunity for stakeholder review and comment on the results of the risk identification, analysis, and evaluation activities. In many cases, the most effective use of time is for the project manager and/or risk analyst to identify, analyze, and evaluate risks independently and then submit the results of the activities to the appropriate stakeholders for review. Review comments may identify the need to repeat some or all of the previous risk analysis activities.

3.4.1.2.3 Plan Risk Aversion

In the Plan Risk Aversion activity (1.1.2.3), the high-level cost and schedule for each risk is estimated and evaluated, the best risk aversion strategies are determined for each. The risk aversion strategies are documented along with an RMP that documents the way the risk aversion strategies will be carried out. Ideally, the appropriate stakeholders are involved in this activity to help evaluate system effectiveness and recommend the best risk aversion strategies.

It is possible to perform risk reduction for extremely high risks through a special risk reduction increment. If a risk reduction increment is undertaken, it is important that any other system development that depends on the results of the risk reduction activities is not planned to start until after the risk reduction activities are complete.

The results of this activity should be documented in an update of the draft RMP.

3.4.1.2.4 Commit to Strategies

The Commit to Strategies activity (1.1.2.4) is a mechanism for formally informing all stakeholders of the contents of the RMP. During this activity, the effectiveness of the risk mitigation strategies and quality assurance plans is assessed. As a result, the consensus and commitment to the risk aversion strategies recommended for each development alternative and high-priority risk is reached. If consensus is not reached or commitment is not secured, or if the risk aversion strategies committed to are not the ones recommended, then the risk analysis activities may need to be repeated, as appropriate.

If modifications are made to the draft RMP, all changes and the rationale for the changes, are documented and distributed to all stakeholders.

3.4.1.3 Plan Increment Development

Figure 22 contains the decomposition of the Plan Increment Development activity (1.1.3). This activity executes the risk aversion, reviews development alternatives for the increment being planned, creates the increment plan, and gets commitment to that plan.

The Execute Risk Aversion activity (1.1.3.1) executes the risk aversions strategies that were planned in the Plan Risk Aversion activity (1.1.2.3). This might include things like prototyping, simulations, studies, or surveys. The RMP is updated to reflect the results of executing the risk aversion strategies.

The Review Development Alternative activity (1.1.3.2) takes the results of the Execute Risk Aversion activity and uses these results to select a development alternative for the system increment being developed. The stakeholders are again part of the selection of the development alternative to ensure stakeholder commitment.

The Create Draft Increment Plan activity (1.1.3.3) defines a plan, including schedule and resources, for developing the current system increment. During this activity, the activities, methods, and tools used to carry out the increment will be selected.

The Commit to Plan activity (1.1.3.4) validates the increment plan. If the plan requires modifications, it is returned to the Create Draft Increment Plan activity(1.1.3.3). Ensuring stakeholder commitment and buy-in is an important part of the Commit to Plan activity.

3.4.1.3.1 Execute Risk Aversion

The Execute Risk Aversion activity (1.1.3.1) executes the risk aversion strategies. These may include prototyping, simulation, surveys, comparative evaluations, evolutionary development, and other appropriate techniques. Complicated or long-term risk aversion activities may be performed in their own development increments. The results of the risk aversion activities impact the selection of development alternatives for this increment. The RMP is updated with all relevant risk aversion execution results.

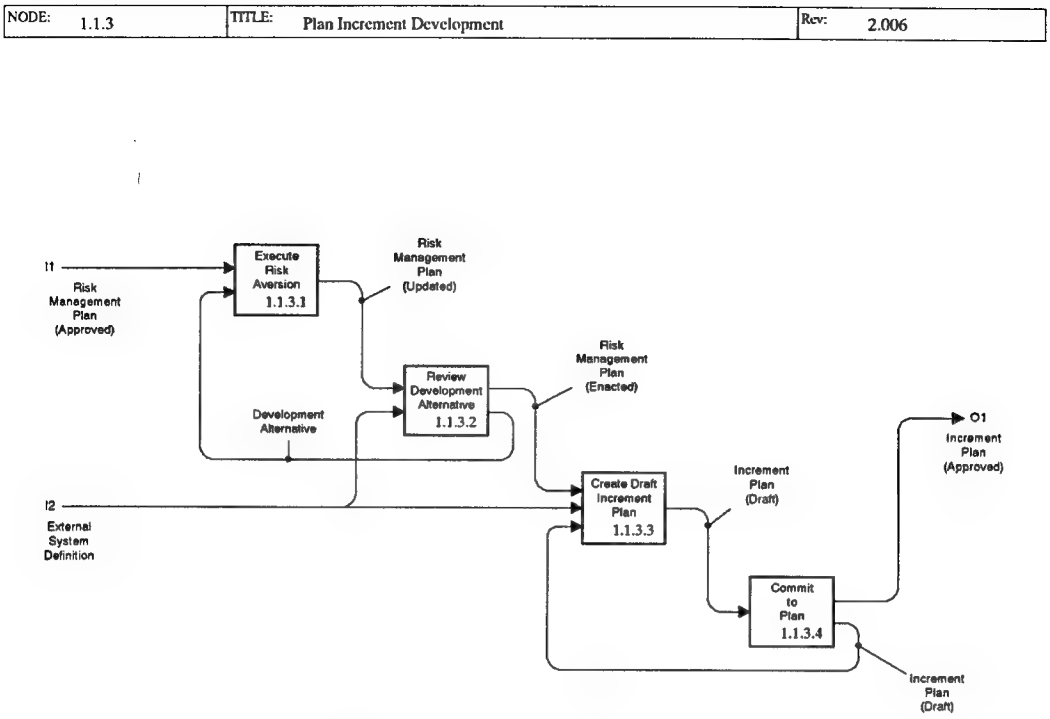


Figure 22. Plan Increment Development

3.4.1.3.2 Review Development Alternative

In the Review Development Alternative activity (1.1.3.2), the results of the Execute Risk Aversion activity are reviewed by appropriate stakeholders. Stakeholder commitment is solicited for the development alternative or approach that is selected based on the results of executing the risk aversion strategies. If the development alternative selected as a result of the risk aversion activities is not committed to by senior management, then additional risk aversion actions are required.

3.4.1.3.3 Create Draft Increment Plan

The main output of the Create Draft Increment Plan activity (1.1.3.3) is an enactable plan for developing the increment. The project's process definition must address increment objectives, development success criteria, constraints, development alternatives, risk mitigation strategies, and any other process driver.

In this activity, technical activities are identified, defined and ordered, and methods, practices, or tools for each task are specified. Activity definitions should include sufficient interfacing events (e.g., IPTs, management reviews, in-process reviews) to coordinate this increment's activities with other increments as well as to satisfy customer control and monitoring needs. Ongoing support activities, such as configuration control, quality assurance, and documentation, are also be included.

Although the basic management functions (planning, monitoring, and controlling) are performed continuously, resources for specific increment development and development support activities are

identified, organized, and allocated after the increment risks are averted. The main output of the Create Draft Increment Plan activity (1.1.3.3) is an increment plan for the development of the product (or part of the product) that will be produced in the current increment, including a specification of the development process that will be used.

The increment plan can be documented as part of the system planning documents or it can be a standalone document. The plan should:

- Establish development goals and associated development success criteria that support the current increment objectives
- Estimate size and scope for the development to be accomplished in the current increment, including:
  - The number and type of product components (e.g., modules, documentation) to be produced as a result of enacting the increment plan
  - The size of each product component
  - The number, severity, and source of errors likely to be found during verification activities
- Identify the activities or methods to be performed in the current increment
- Define/redefine activities, methods, and supporting work products, if necessary
- Sequence the activities if not sequenced by the selected method
- Estimate development cost and schedule for each increment activity and allocate resources
- Select and allocate supporting tools
- Define work packages, or the lowest work breakdown structure (WBS) level, for the key activities defined for the current increment

In addition, the plan should:

- Address any customer requirements
- Include ongoing support activities in the increment plan, such as configuration management, quality assurance, and documentation
- Comply with customer policies, procedures, standards, and regulations, if necessary
- Comply with the organizational process definition, if available and appropriate
- Use any available historical planning and engineering data for estimating purposes
- Take advantage of organizational methods, tools, training, and support

3.4.1.3.4 Commit to Plan

The Commit to Plan activity (1.1.3.4) is an opportunity for stakeholders to review and comment on the results of the increment planning and scheduling activities. Consensus that the activities of the increment plan are appropriate to meeting increment objectives must be reached. The increment plan may be updated according to stakeholders' comments. When commitment to the detailed increment plan is secured, a formal briefing to senior management on the current increment development activities can occur. This briefing may be a natural extension of any existing, periodic internal management reviews, or it may be scheduled separately. As a result of committing to the increment plan, work packages are opened and assigned and the detailed increment process, as documented in the increment plan, is considered enactable.

3.4.1.4 Track Increment Development

Figure 23 contains the decomposition of the Track Increment Development (1.1.4) activity. This activity monitors and reviews the progress of the increment, makes minor updates to the increment plan, if necessary, and reviews the system increment work products being produced.

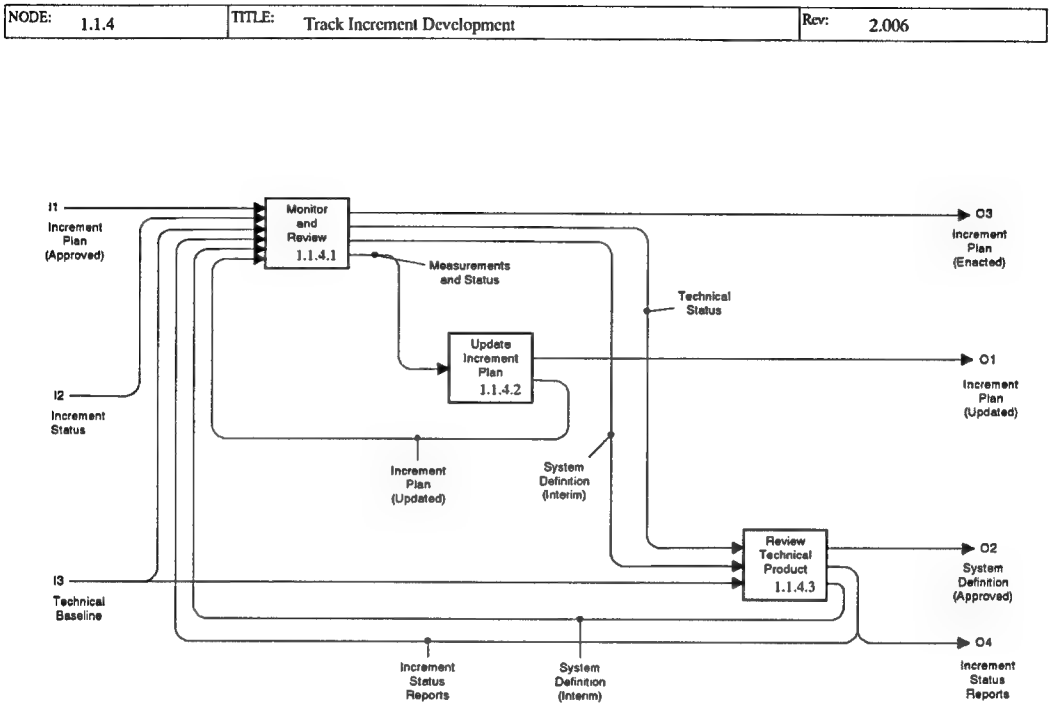


Figure 23. Track Increment Development

The Monitor and Review activity (1.1.4.1) assesses the progress of the increment development.

The Update Increment Plan activity (1.1.4.2) uses the assessed progress that was calculated during the Monitor and Review activity (1.1.4.1) to update the increment plan.

The Review Technical Product activity (1.1.4.3) evaluates the system definition against the increment success criteria. If the system definition requires modification, the updates are again monitored and reviewed in the Monitor and Review activity (1.1.4.1). If the system definition remains incorrect or incomplete, then a new increment may be necessary. In this case, the Develop System Plan (1.1.5) is initiated, lessons learned from this increment are documented, and planning for the next increment proceeds.

3.4.1.4.1 Monitor and Review

The Monitor and Review activity (1.1.4.1) maintains management control over the technical development and ensures that the technical activities are carried out in accordance with the management plan. This activity periodically captures and analyzes the status of the increment to provide management with insight into the cost, schedule, process fidelity and fitness, and technical

performance status of the increment. Raw activity progress/status is analyzed to produce management metrics that support decision making regarding corrective action. Work package status for each cost account is collected, analyzed, and summarized to the highest schedule level. For each parameter selected for monitoring, actual and projected performance is updated, trends are analyzed, and unfavorable trends or values are noted. The following should be reviewed and analyzed based on the data and measurements collected during the enactment of the technical activities that make up Define System Increment (see Figure 12, page 28):

- Product component size status
- Product component error data
- Product component change data
- Increment schedule status
- Increment cost status
- Increment risk status
- Increment quality assurance status

Periodic increment reviews are conducted as a part of this activity. These reviews serve two purposes. First, they demonstrate to the team and other stakeholders that the current increment and overall system development are under control. Second, they provide a mechanism to obtain a commitment to proceed with the increment plan. At these reviews, decisions can be made to reallocate resources, replan schedules, or reassess risks for the current increment activities. All such changes are reflected in an updated increment plan and/or evolved increment process.

The *Software Measurement Guidebook* (Software Productivity Consortium 1992) provides information on specific size, cost, schedule, and error measurement and analysis methods.

3.4.1.4.2 Update Increment Plan

The Update Increment Plan activity (1.1.4.2) makes minor modifications (i.e., no perceived risk) to the increment plan to account for technical and managerial problems and situations. For example, if a particular tool is unavailable when scheduled, this activity may decide to either postpone the task that requires the tool or substitute another tool. If major modifications (i.e., perceived risk) to the plan are necessary, then the Develop System Plan activity (1.1.5) is initiated so that all impacts of the required modifications to the increment development can be assessed relative to the entire system development effort.

3.4.1.4.3 Review Technical Product

During the Review Technical Product activity (1.1.4.3) a technical product review is performed. The technical product review is a stakeholder review of the product (or part of the product) developed in the current increment. The purpose of this activity is to ensure that increment objectives and development success criteria have been met. This review is an opportunity for stakeholder review; the ongoing Monitor and Review activity, however, is responsible for identifying and planning to resolve any problems with meeting increment objectives or development success criteria. The Increment Plan is only updated here when risk analysis is deemed unnecessary. If it is not clear that risk analysis is unnecessary, then the management team should cycle through the Manage Development Effort activities (1.1) (see Figure 19).

3.4.1.5 Develop System Plan

Figure 24 contains the decomposition of the Develop System Plan activity (1.1.5). This activity implements product change control, reviews progress of the technical development effort, updates the long-term system plan for the foreseeable future, and gets commitment to proceed with the updated system plan.

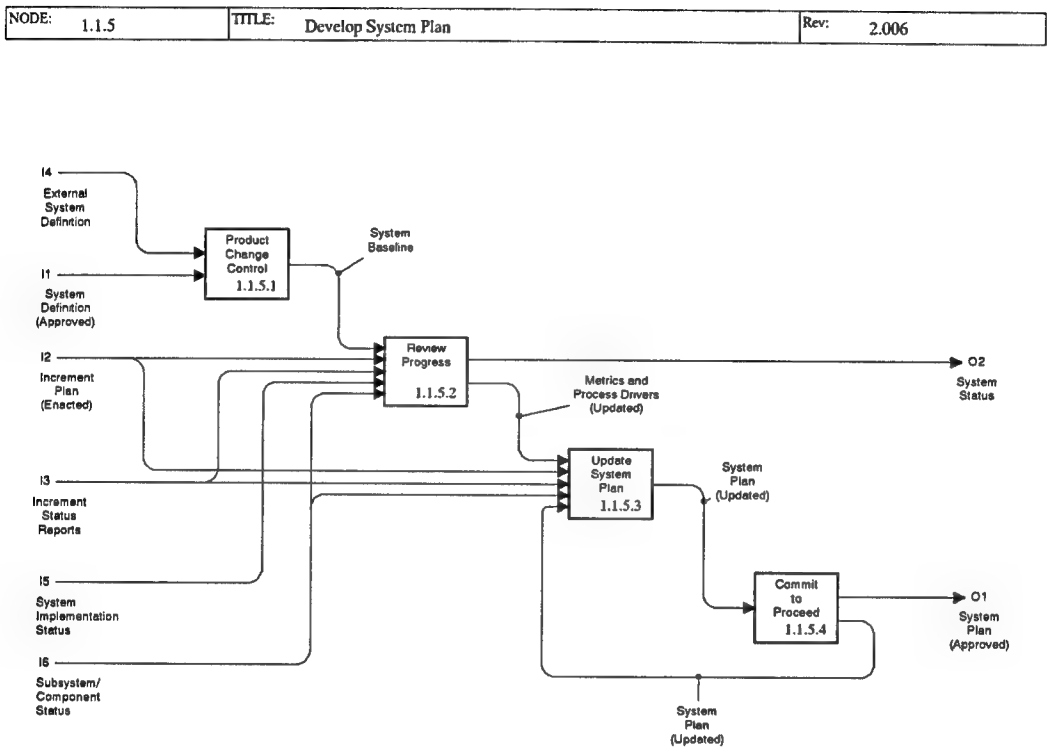


Figure 24. Develop System Plan

The Product Change Control activity (1.1.5.1) baselines any externally deliverable system definition created in this increment.

The Review Progress activity (1.1.5.2) assesses the progress of the system increment (e.g., compares the increment's system definition against the increment's development success criteria) including the progress of any of this system's constituent subsystems or components. The system status is also collected, and the appropriate information is summarized to update the system metrics and identify additional process drivers.

The Update System Plan activity (1.1.5.3) uses the metrics and process drivers that result from the progress review of this system increment, as well as the status of the higher level system of which this system is a part (if applicable), to update the system plan.

The Commit to Proceed activity (1.1.5.4) will validate the system plan. If the plan is found lacking, it may be updated. This activity will also get the stakeholders to buy in to the system plan.

3.4.1.5.1 Product Change Control

During the Product Change Control activity (1.1.5.1) the product (or part of the product) resulting from any externally deliverable increment is baselined. The implementation of changes is tracked to control products to ensure that the configuration of the product is controlled at all times. Each baseline is established by identifying the configuration units that constitute the baseline. All subsequent changes are tracked in the configuration status. The history of changes to each configuration item is maintained throughout the system development. All items are stored under control in a secure, limited-access repository.

3.4.1.5.2 Review Progress

In the Review Progress activity (1.1.5.2), actual measures are evaluated against those estimated in the increment plan, development success criteria are examined to ensure that they were met, and lessons learned are identified. An analysis of how the final increment status may affect the process for the rest of the system development is conducted. Specifically, increment data is reviewed and verified against the actual measurements.

3.4.1.5.3 Update System Plan

During the Update System Plan activity (1.1.5.3), the system planning documents are updated to reflect the lessons learned from the increment. The main output of the Update System Plan activity is the updated system plan. The updated system plan includes new process drivers that were identified during the analysis of the increment's development progress. The system plan is also updated to reflect the current understanding of system objectives, constraints, development alternatives, and risk mitigation strategies, and to incorporate the legacy inherited from all enacted increments.

3.4.1.5.4 Commit to Proceed

A key concept during the Commit to Proceed activity (1.1.5.4) is the commit concept, which comes into play at the end of the increment. The commit concept is one of the most important in using the GSEP management activities. It is similar in purpose to a baseline; all stakeholders are briefed on the results of the current increment and on any changes in system development plans and agree with the decisions made regarding what to do next. Stakeholders include project management at all levels and the IPTs. The review may also include customer representatives. The purpose of the review is to determine whether system-level objectives, alternatives, and constraints are still feasible, to agree on the objectives for the next increment, and to commit resources to that increment. The system planning documents may be modified as the stakeholders reach agreement. All changes should be documented with the change rationale and distributed to all stakeholders.

3.4.2 Define System Increment

Figure 25 contains the decomposition of the Define System Increment activity (1.2). This activity is the technical subactivity in the GSEP that analyzes the needs the system must address, defines the system requirements, defines the functional architecture that provides an allocation-independent design that meets the system requirements, synthesizes allocated architectures that meet the system requirements, evaluates each of the alternatives and selects the most desirable, validates and verifies the technical work products, and controls the technical baseline.

The Evaluate Alternatives (1.2.5) and the Validate and Verify Solution (1.2.6) activities are tightly coupled. The Evaluate Alternatives activity is responsible for the optimization of various systems designs and the trade-off of those designs against each other. During the Validate and Verify Solution activities, the intent is to ensure that the optimized work products do what they are supposed to do. In verification, what the verified work product supposed to do is interpreted by the preceding activity(s) work products. In validation, what the design is supposed to do is interpreted by the user/customer.

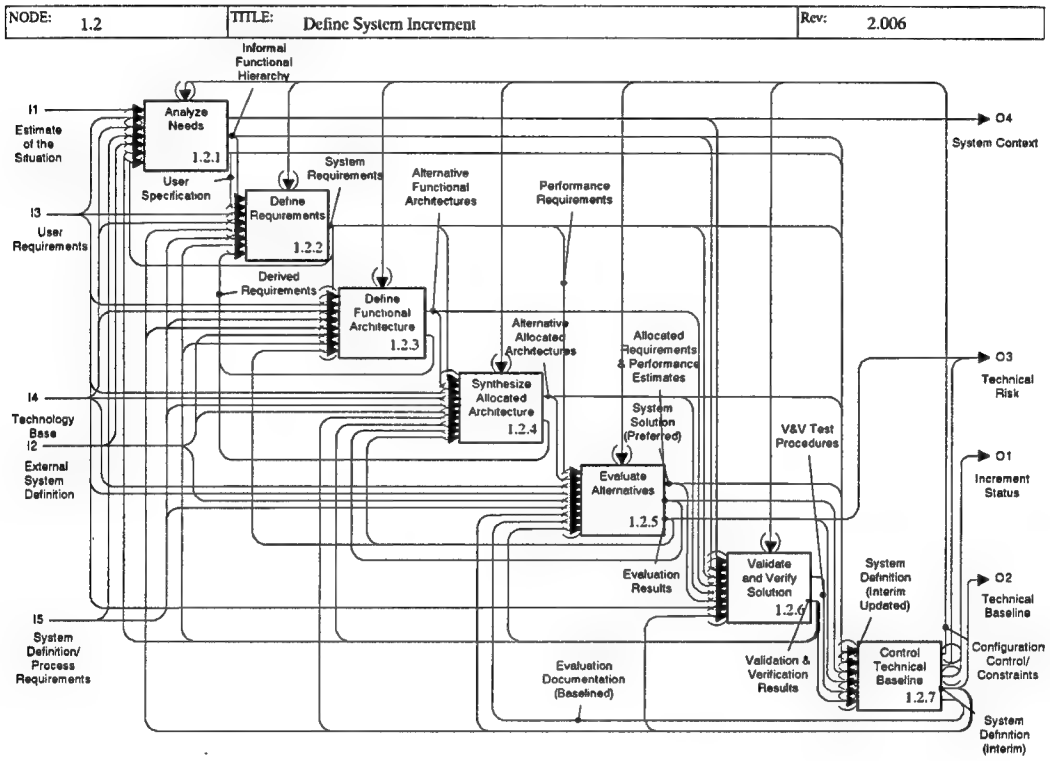


Figure 25. Define System Increment

The Analyze Needs activity (1.2.1) determines the stakeholders, assesses the problems the system is to solve and the needs that the system is to address, defines the environment in which the system is to operate, and develops the informal functions the system is to perform. The goal of Analyze Needs activity is to define the true needs of the users and customers by fully understanding the solution stakeholders (who they are, what they really need, and why they need it) and identifying the potential environments in which a system solution will be manufactured, operated, and maintained.

The Define Requirements activity (1.2.2) takes the informal functional hierarchy produced by the needs analysis and defines the requirements that the system must satisfy to be acceptable to the user and customer of the system. The requirements are used to stipulate the conditions (predicted, or legally binding) under which the system will be purchased/procured by the customer.

The Define Functional Architecture activity (1.2.3) creates a functional architecture made up of functions and interfaces. In this activity, an allocation-independent design of the system is created. The top-level functions are defined by partitioning the system requirements. These functions must

meet allocation constraints that must be considered in the creation of the system (e.g., political considerations [subsystems that need to be developed in appropriate states], family of hardware configurations [existing hardware that is likely to be used for system implementation]).

The Synthesize Allocated Architecture activity (1.2.4) uses the system requirements, the alternative functional architectures, and input from the user/customer to produce the alternative allocated architectures.

The Evaluate Alternatives activity (1.2.5) assesses the various alternative allocated architectures, performs sensitivity analysis, allocates technical parameters, identifies the technical risks and problems, evaluates the alternately allocated architectures against each other, and selects the preferred system solution.

The Validate and Verify Solution activity (1.2.6) defines the test procedures for verifying and/or validating a work product. This activity also performs these test procedures to assess the system's measure of effectiveness against the user/customers stated needs and against the previously generated constraints defined in activities prior to the activity who's work products are being evaluated.

The Control Technical Baseline activity (1.2.7) maintains and versions the technical decision data that gives the rationale for technical decisions that have been made. This activity also maintains the technical work products configuration control.

3.4.2.1 Analyze Needs

Figure 26 contains the decomposition of the Analyze Needs activity (1.2.1). This activity determines the stakeholders, assesses the problems the system is to solve and the needs that the system is to address, defines the environment in which the system is to operate, and develops the informal functions the system is to perform.

The Determine Stakeholders activity (1.2.1.1) identities the system stakeholders. These stakeholders are consulted throughout the definition of the system and, especially, at key decision points. The system stakeholders identified and consulted in this activity are not identical to those identified in the management subprocess. These stakeholders will overlap, but there may be unique stakeholders in each group. As an example, a stakeholder may be identified in the user community for representing the users in the management subprocess, but in the technical subprocess, several users that represent specialists may be identified.

The Assess Problems and Needs activity (1.2.1.2) identifies the problems the system is being created to solve as well as the needs the system is to meet. During this activity, these problems and needs must be classified as to importance. Conflicts between needs and problems must be resolved. Measures of effectiveness and critical influencing factors must be identified and the operational concept defined. The identified stakeholders are used to help identify these needs and problems.

The Define Environment activity (1.2.1.3) identifies the environment in which the system must be manufactured, operated, and supported.

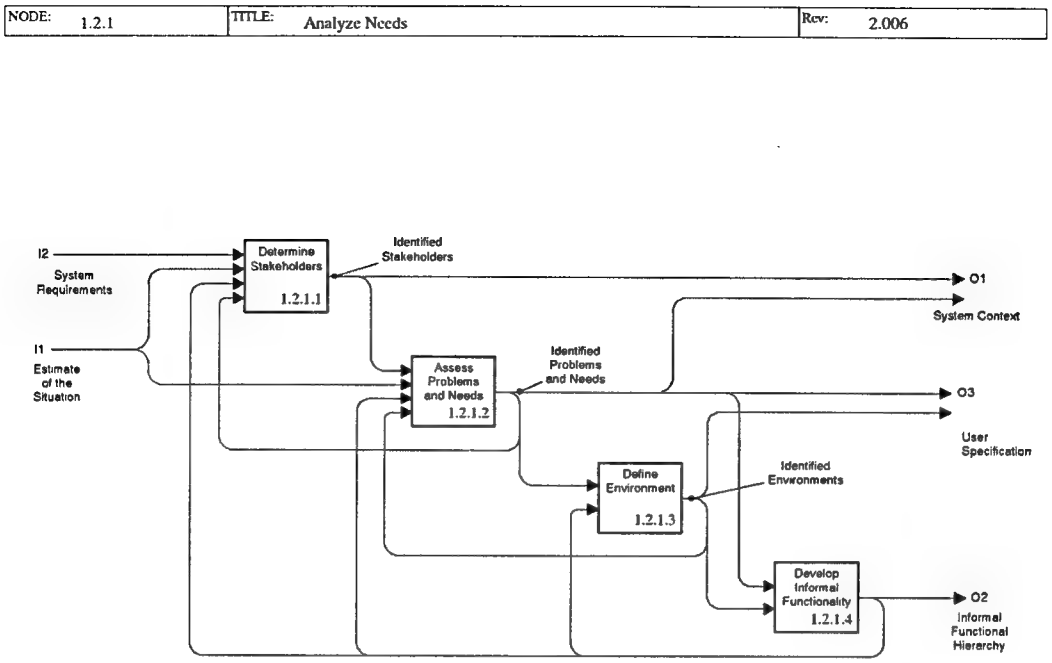


Figure 26. Analyze Needs

The Develop Informal Functionality activity (1.2.1.4) defines an informal functional architecture that defines the system functions at a very high level. These functions should in no way limit or constrain the solution space of the system alternatives. The purpose of these functions is to constrain and bound the problem domain, not to define the anticipated solution.

3.4.2.1.1 Determine Stakeholders

During the Determine Stakeholders activity (1.2.1.1), the system stakeholders are identified based on preliminary needs for the system and its anticipated life cycle. This includes the identification of all elements that are involved in system definition, fabrication, and use (operations and support).

3.4.2.1.2 Assess Problems and Needs

The Assess Problems and Needs activity (1.2.1.2) completely analyzes the user and customer concerns that a system solution will be designed to solve and the operational scenarios that support the system operational concept. Problems and needs must be viewed in terms of "potentials," in that many of the problems will arise in the future or unanticipated needs will arise during system development or use. Measures of effectiveness are defined for assessing the ability of the system to address the identified problems and needs.

The domain of the problem being considered is often larger than the initial stated needs of the user and the customer. The goal of this activity is to identify user and customer problems and needs, known and anticipated, at their most basic level and not strictly a user/customer interpretation of them.

Another objective during this activity is to define the critical influencing factors that drive the technical development of the system.

3.4.2.1.3 Define Environment

During the Define Environment activity (1.2.1.3), the system operational environment is determined. This determination must be based on the problems and needs that have been identified. The environment defines the context of all possible operational scenarios for the system throughout its life cycle.

3.4.2.1.4 Develop Informal Functionality

During the Develop Informal Functionality activity (1.2.1.4), the user needs are decomposed into high-level informal functions that allow restatement of the needs in engineering terms. These functions are very loosely defined but capture the intent of the user needs. Technical performance measures are developed that can be used to determine whether the functions adequately meet the users needs.

3.4.2.2 Define Requirements

Figure 27 contains the decomposition of the Define Requirements activity (1.2.2). This activity takes the informal functional hierarchy and the user specifications produced by the needs analysis and, in conjunction with direct interactions with the customer and user, identifies the requirements that ensure the system will meet the customer and users needs. The technical performance measures defined in the informal functional hierarchy are refined and augmented, and critical performance measures identified. Critical performance measures will be tracked closely in the evaluation, validation, and verification of the system. Behavioral requirements are defined that ensure that the system will generate appropriate responses to system input or temporal events.

Performance requirements are derived from the needs and assessed against their effect on the system's ability to meet the customer/user requirements. These performance requirements include physical requirements (e.g., weight, power consumption) as well as other quantitative requirements (e.g., timing, reliability, maintainability) that define the performance characteristics of the behavioral requirements.

After the behavioral and performance requirements are established, they are merged to define the system requirements by identifying the behavioral requirements (if any) that the performance requirements constrain.

Finally, the requirements are refined by analyzing the consistency and completeness of the requirements as well as their effectiveness in guaranteeing the satisfaction of the customer and user needs.

The Determine Behavioral Requirements activity (1.2.2.1) formalizes the informal functional hierarchy into a set of testable behavioral requirements that define the behavior of the system that is required to produce the mandatory system outputs.

NODE: 1.2.2	TITLE: Define Requirements	Rev: 2.006

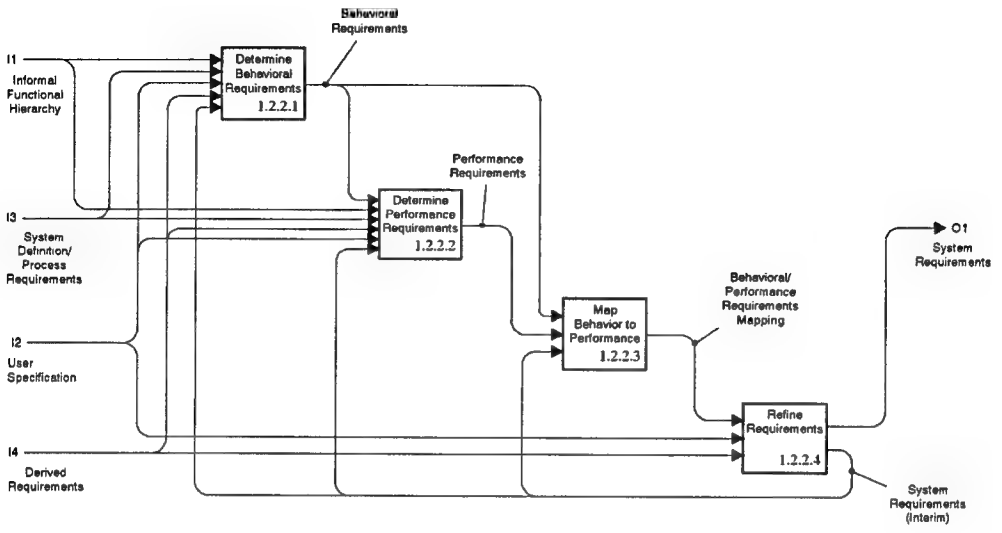


Figure 27. Define Requirements

The Determine Performance Requirements activity (1.2.2.2) formalizes the quantitative aspects of the informal functional hierarchy. The performance requirements define a set of testable performance objectives that define the quantitative aspects of the behaviors the system is to perform. These quantitative aspects define physical characteristics and define the performance characteristics of the behavioral requirements.

The Map Behavior to Performance activity (1.2.2.3) correlates the performance requirements to the behavioral requirements that are affected by the performance requirements.

The Refine Requirements activity (1.2.2.4) refines the requirements to address derived requirements discovered during the association of the performance and behavioral requirements. This activity also ensures that the system requirements are testable.

3.4.2.2.1 Determine Behavioral Requirements

The Determine Behavioral Requirements activity (1.2.2.1) starts with an initial set of functions, generated from the customer and user inputs, and determines the behavioral requirements for the system. These requirements must be spawned from each of the identified informal functions and should not dictate a design solution. During the identification of the behavioral requirements, the technical performances measures are refined, augmented, and associated with the relevant behavioral requirements. The behavioral requirements do not specify performance requirements such as timing performance or weight. These requirements establish the data requirements that specify the information the system is to produce.

3.4.2.2.2 Determine Performance Requirements

During the Determine Performance Requirements activity (1.2.2.2), the user needs are assessed and performance characteristics that are mandated in order to meet those needs are defined. These performance requirements are used to constrain the implementation of the behavioral requirements and the physical system characteristics. The performance requirements include requirements that define the required physical characteristics of the system, such as weight, as well as characteristics of the behavior that define how well the behavior must be carried out. All of the characteristics must be quantifiable and testable. The characteristics include things such as accuracy of results, the efficiency of the system, or the speed with which the system must accomplish the behavior. During the identification of the performance requirements, the technical performances measures are refined, augmented, and associated with the relevant performance requirements.

3.4.2.2.3 Map Behavior to Performance

During the Map Behavior to Performance activity (1.2.2.3) the performance requirements are mapped to the behavioral requirements they constrain. Often, the performance requirements are related to particular behaviors in the system. Other times, the performance requirements are written against the system as a whole and will not map to any individual behavior.

3.4.2.2.4 Refine Requirements

After initial requirements are determined, the Refine Requirements activity (1.2.2.4) evaluates them to ensure that they map to identified functions and that the requirements are testable. Once established, refined, and baselined, these requirements serve as the foundation upon which the system is defined. The Refine Requirements activity interacts with other technical activities that identify derived requirements. These derived requirements must pass through refinement before they are accepted and baselined. The activity must also ensure that the "derived" requirements are necessary and sufficient to meet the objectives of the original requirements. The customer must confirm that the requirements satisfy their needs and expectations. Derived requirements are not based on design decisions; instead they are based on implications of other requirements.

In this activity, the requirements are also assessed against their impact on the system's measure of effectiveness. The sensitivity of the performance requirements to the resultant system's measure of effectiveness is assessed, and the performance requirements may be refined based on this sensitivity.

3.4.2.3 Define Functional Architecture

Figure 28 contains the decomposition of the Define Functional Architecture activity (1.2.3). This activity defines functions based on a partitioning of the system requirements. The partitioning of the system is based on partitioning criteria that include performance and design considerations. These functions are based on design decisions that begin to focus on a solution. As such, the functions are now constrained to be those that the systems engineer believes will result in feasible solutions based on reasonable implementation environments. This activity also determines the interfaces, in terms of data structures, that must exist between the functions in order to meet the required system behaviors.

As the system functions are identified, alternate solutions that meet the requirements are identified. These solutions often demand a lower level decomposition of the system functions that define the

intended solution. These lower level functions are identified and defined in the functional architecture. In some cases, interfaces are not cleanly defined. This may require a repartitioning of functions or functional overlap, where the same function appears to be occurring in several areas or at various levels of system decomposition.

The development of the alternative functional architectures does not result from a decomposition of the informal functions. Instead, the alternative functional architectures are based on the specified system requirements. The functions are not constrained by the informal functions defined earlier. This distinction is made to ensure that early assumptions and biases do not influence the system design. As an example, assume that a navigation function is an informal function in the system. It may be that there are multiple lower navigation functions that are better combined with vehicle control functions and other navigation functions that are better combined with communication functions. This may result in an autopilot function and a communication function that are not a decomposition of the navigation functions but functions that were defined from analysis of the system-level requirements instead.

The alternative functional architectures could be defined using object-oriented or structural decomposition techniques. The goal is to define the functions (or methods) that the system will perform as well as the information that is needed to perform them.

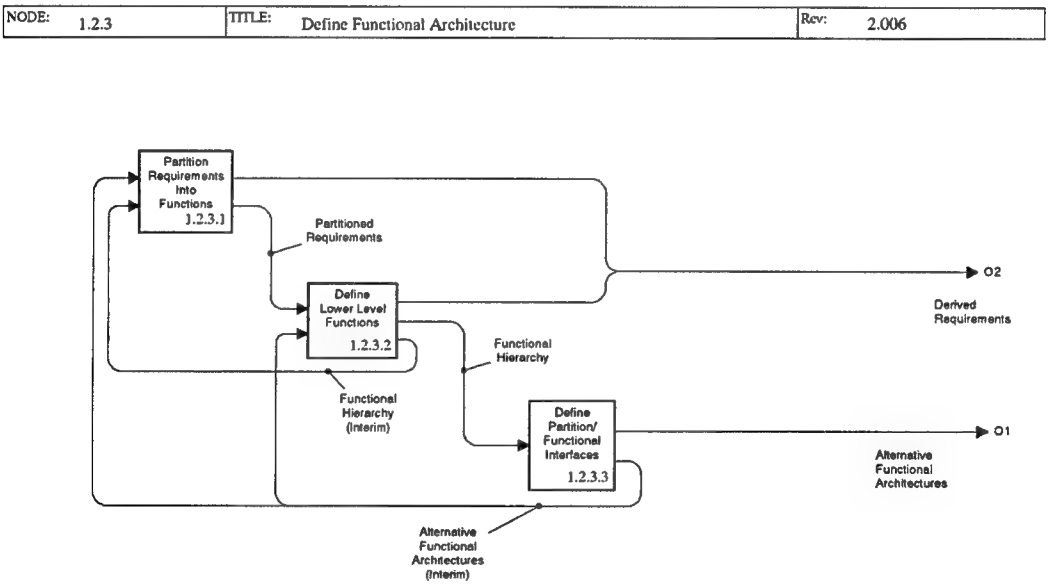


Figure 28. Define Functional Architecture

The Partition Requirements Into Functions activity (1.2.3.1) takes the system requirements and partitions them into a set of functions that define the allocation-independent system design. These functions are not necessarily mapped neatly to the informal functions defined earlier but are directly derived from the system requirements. The partitioning is based on the behavioral requirements but influenced by the performance requirements.

The Define Lower Level Functions activity (1.2.3.2) decomposes the functions defined from the partitioned system requirements into lower level functions. These functions are defined as the result of design decisions that begin to specify how the system will satisfy the system requirements. These design decisions should still be allocation-independent but begin to constrain the allocation decisions.

The Define Partition/Functional Interfaces activity (1.2.3.3) defines the interfaces between the partitions and functions that compose the functional architecture. In this activity, the information that flows between the functions is defined as well as the behavior of the interfaces.

3.4.2.3.1 Partition Requirements Into Functions

During the Partition Requirements Into Functions activity (1.2.3.1), the system requirements are partitioned into a set of functions that define the allocation-independent system design. The partitioning of functions is based on the degree of interaction between the behavioral requirements as well as the mapping of performance requirements to the behavioral requirements. Another influencing factor in the partitioning of the system requirements into functions is design constraints. If there are reuse requirements, or commercial off-the-shelf (COTS) systems that must be used, some of the partitioning may be based on these decisions. Finally, there may be political considerations that constrain the way the requirements are partitioned.

3.4.2.3.2 Define Lower Level Functions

The Define Lower Level Functions activity (1.2.3.2) decomposes the functions defined from the partitioned system requirements into lower level functions. The lower level functions begin to define methods and techniques for accomplishing the system requirements. The lower level functions are not intended to define an allocated solution but are intended to define a logical architecture that specifies a set of interacting functional entities that satisfy the system requirements. The lower level functions specify constraints on the allocation that further limit the possible solution, but there is still a range of solutions that could be defined to satisfy the functional architecture.

3.4.2.3.3 Define Partition/Functional Interfaces

The Define Partition/Functional Interfaces activity (1.2.3.3) identifies and describes the interfaces between the functions identified in the functional hierarchy. In this activity, each of the functions is assessed to determine the information that is used to produce the output of the function. From this analysis, the information flows between functions and to the environment are identified. The behavior of the interfaces is also specified (e.g., whether the data is buffered, the anticipated frequency of the external data flow, and the dependence of the functions [e.g., a function requests or responds to data]).

3.4.2.4 Synthesize Allocated Architecture

Figure 29 contains the decomposition of the Synthesize Allocated Architecture activity (1.2.4). During this activity, the alternative functional architectures are allocated to alternative allocated architectures. The allocated architectures are potentially physical architectures that define the viable alternatives. This may result in the physical layout of the system, or only the logical subsystems, and their interactions. If the solution defines a physical architecture, then the functions are allocated to hardware, software, and people (procedures). Interfaces are defined that communicate the interactions between the parts in the allocated architecture, and technical parameters are defined that

drive the performance parameters of the allocated parts. Together, the allocation, interfaces, and technical parameters define the allocated architecture.

In this activity, the Integrate Design activity (1.2.4.4) calls out the integration of the subsystem and component designs that define the system. This design integration allows reevaluation of system designs using more accurate information provided by the lower level designs. This means that design problems can be found much earlier in the system definition process saving both time and money. This activity is also used to communicate across the system to subsystems and components that have a direct bearing on the system under consideration. As an example, the drive shaft must be producible by the drive shaft pressing machine. As a result, the the drive shaft pressing machine must provide input to the Integrate Design activity.

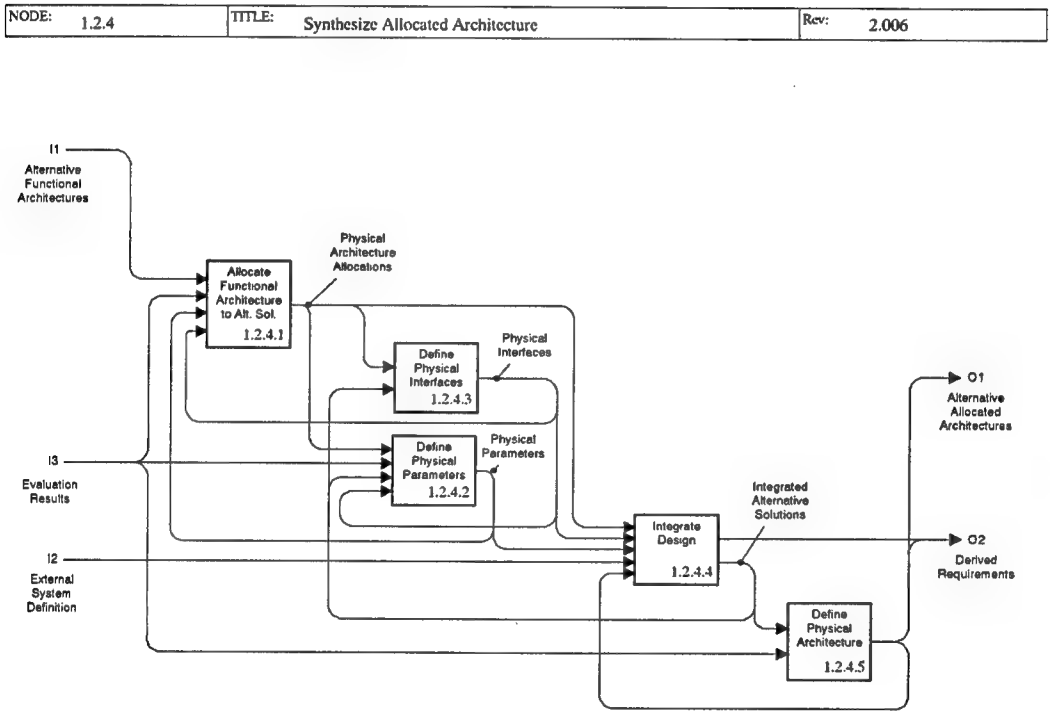


Figure 29. Synthesize Allocated Architecture

During the Allocate Functional Architecture to Alternative Solutions activity (1.2.4.1) the alternative functional architectures, which consist of the functional decomposition, hierarchy, and interfaces, are allocated to potential system solutions. Recognizing that functions can be accomplished via different means, each alternative solution allocates the functionality to different implementing mechanisms.

The Define Physical Parameters activity (1.2.4.2) defines the algorithms and parameters that drive the architecture that is selected. These parameters define acceptable ranges of values that can be used to drive the performance of an assembly or subassembly.

The Define Physical Interfaces activity (1.2.4.3) identifies the actual physical interfaces that are needed to support the logical interfaces defined by the mapping of the formal functions and interfaces to the alternate solutions.

The Integrate Design activity (1.2.4.4) takes the "as built" designs (if they exist) from the subsystems or components that are created in lower level or peer system definitions and uses the results to update the system architecture.

The Define Physical Architecture activity (1.2.4.5) takes the architecture defined by the allocation, interfaces, and parameters and addresses design considerations that result from the chosen solution.

3.4.2.4.1 Allocate Functional Architecture to Alternative Solutions

The Allocate Functional Architecture to Alternative Solutions activity (1.2.4.1) defines multiple system architectures that seem viable. These architectural allocations define alternate physical architectures and the logical interactions between the allocated functions. Candidate mechanisms for carrying out the functions can be identified from the technology base (in the case of off-the-shelf systems). The mechanisms are partitioned into assemblies or subassemblies based on the functional architecture and the allocated performance characteristics.

3.4.2.4.2 Define Physical Parameters

The Define Physical Parameters activity (1.2.4.2) defines specific characteristics of an identified logical or physical entity. Physical entities include hardware, software, or people (procedures). The technical parameters drive the performance of the logical or physical entities. The characteristics identified include performance, size, mass, and timing.

Physical parameters drive algorithms that define the system performance (see Figure 30). These parameters are used to predict and evaluate whether the system design being assessed will meet the system performance requirements. It is often the case that these parameters continue to be refined throughout the life of the system.

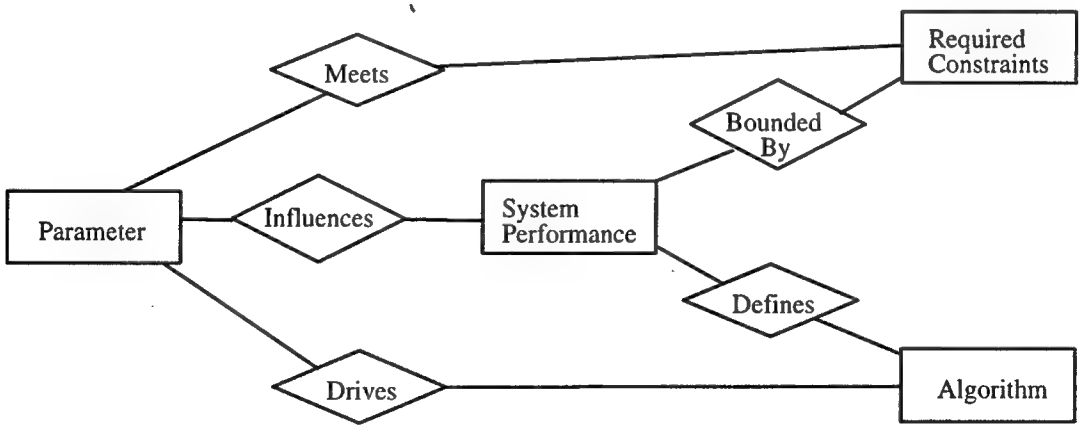


Figure 30. Defining System Parameters

3.4.2.4.3 Define Physical Interfaces

Following identification of the potential parts of the system, the Define Physical Interfaces activity (1.2.4.3) determines the necessary interfaces between the parts of the system. The interfaces may be

physical interfaces and could be as simple as placement of components in a subsystem, or as complex as determining the best approach to providing for multiple functional interfaces through physical means, possibly in a distributed fashion. The physical interfaces define the data structures and standards used to communicate the data.

3.4.2.4.4 Integrate Design

The Integrate Design activity (1.2.4.4) integrates the actual system definition from all lower level designs within its purview. If a subsystem has allocated parameters to lower level components, this activity takes the actual design information and incorporates it into the subsystem design.

3.4.2.4.5 Define Physical Architecture

During the Define Physical Architecture activity (1.2.4.5), each solution is defined to a level of actual physical and/or concrete characteristics. This is basically a definition of how the functional architecture will be performed. In defining a physical architecture, decisions must be made as to how functions will be performed, including preliminary determination as to a mechanical, electrical, software, or human solution. The resultant architecture may introduce other functionality that is created to deal with the consequences of the allocation. This includes the identification of failure mechanisms and other entities identified to support startup, restart, reconfiguration, and degraded operation.

3.4.2.5 Evaluate Alternatives

Figure 31 contains the decomposition of the Evaluate Alternatives activity (1.2.5). This activity assesses the system, performs sensitivity analysis, allocates technical parameters, identifies and assesses technical risks and problems, identifies and performs trade-offs, and selects the system solution.

The assessment of the system is separate from the sensitivity analysis and trade-offs. The assessment activity assesses its various performance characteristics. During the sensitivity analysis, an attempt is made to optimize the system performance characteristics for a particular system design. In trade-off analysis, alternate optimized system designs are evaluated against each other.

The Assess System activity (1.2.5.1) defines models to be used to evaluate the system as a whole. During this activity, each of the performance analysis disciplines (the "ilities") build models to assess the system against the performance characteristics that affect that discipline.

The Perform Sensitivity Analysis activity (1.2.5.3) evaluates the system with respect to variations in system parameters and external environment interactions. This evaluation may affect the assessment of technical risks and problems.

The Allocate Performance to Technical Parameters activity (1.2.5.4) defines the values of the technical parameters for the subsystems/components/elements that define the actual selected allocated architecture. The analysis that defines the performance of the system against the critical characteristics within the evaluating discipline occurs in this activity.

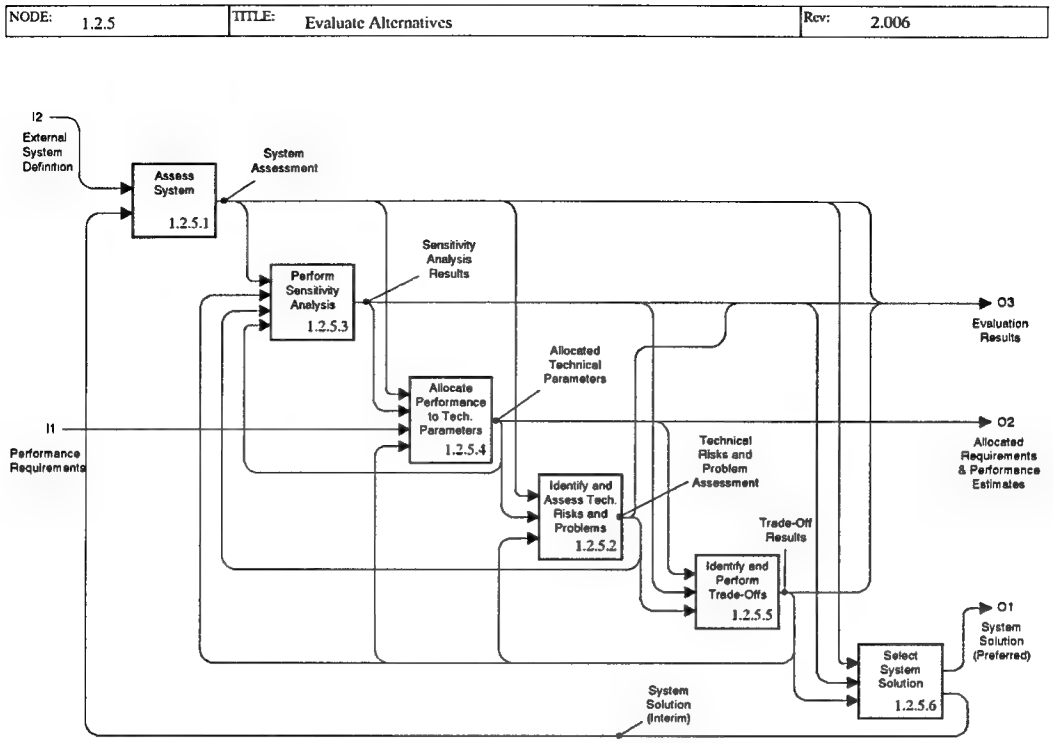


Figure 31. Evaluate Alternatives

The Identify and Assess Technical Risks and Problems activity (1.2.5.2) takes the system assessment and identifies the system's problems and technical risks. The evaluation results are used to identify the technical risks associated with the system solution.

The Identify and Perform Trade-Offs activity (1.2.5.5) is extremely broad in nature. It performs an analysis to determine the best of multiple potential solutions.

During the Select System Solution activity (1.2.5.6), the system trade-off results are evaluated to select the preferred solution.

3.4.2.5.1 Assess System

The Assess System activity (1.2.5.1) assesses the entire system that is being developed. Under normal circumstances, it is impractical to continuously analyze the full scope of the system, so analysis is performed on subsystems or components only. In this activity, a total picture must be taken in order to ensure the analysis of all elements of the system, many of which will not fit neatly into a subsystem or component element. In many cases, it is this system assessment that identifies system weaknesses or missed interfaces. It is preferable that any system assessment be based on a well defined set of technical performance measures (TPMs).

In this activity, the specialty evaluation disciplines are exercised to create models used to evaluate the performance of the system in that domain. This may include air vehicle lift characteristics, reliability, and reusability. Many times the evaluated solution defines parameters that are used along with parameters external to the system in algorithms defined for the domain in question. The goal of the

activity is to capture and understand all of the issues that are important to each of the disciplines that are critical to the system.

3.4.2.5.2 Perform Sensitivity Analysis

The Perform Sensitivity Analysis activity (1.2.5.3) analyzes the system's sensitivities to varying environments and design parameters, i.e., the impact on system capability due to a change of design functionalities or variabilities in system concepts.

3.4.2.5.3 Allocate Performance to Technical Parameters

During the Allocate Performance to Technical Parameters activity (1.2.5.4), technical parameters, including all physical characteristics and performance/operating requirements, are defined for the allocated architecture. This allocation of characteristics and capability is determined through analysis techniques that attempt to partition system characteristics in a logical and optimal fashion. The result is a large number of orthogonal evaluation results that all summarize the performance and behavior of the system in question. Any external system definitions or implementation results can be used in this evaluation to get a more accurate evaluation.

The evaluation results are correlated against relevant technical performance measures for the system. The goal is to optimize the alternative architecture against the technical performance measures to give the most desirable performance characteristics attainable by the alternative allocated architecture under consideration.

3.4.2.5.4 Identify and Assess Technical Risks and Problems

The Identify and Assess Technical Risks and Problems activity (1.2.5.2) determines technical risks and problems and evaluates those risks to determine their potential impact to the system. The identified risks and problems can then be mitigated. As a result of this activity, key design issues are identified that must be resolved to produce a successful system.

3.4.2.5.5 Identify and Perform Trade-Offs

The Identify and Perform Trade-Offs activity (1.2.5.5) identifies the viable system alternatives and completes a trade-off analysis of the systems in question. Each system is assessed against the technical performance measures, the level of risk, and the problems associated with the system. The trade studies done assess the effectiveness of a particular solution against the system measures of effectiveness and communicate information back to the Allocate Performance to Technical Parameters activity (1.2.5.4) to optimize the solution along the assessment parameters.

3.4.2.5.6 Select System Solution

The Select System Solution activity (1.2.5.6) is an analysis to determine the optimal solution among many alternatives. This may be an analysis to select an individual alternative, or it may be an analysis to consider the melding of various alternatives into a new integrated alternative.

3.4.2.6 Validate and Verify Solution

Figure 32 contains the decomposition of the Validate and Verify Solution activity (1.2.6). This activity defines the test procedures that are used to verify and/or validate a work product. This activity also

implements these test procedures to actually verify and/or validate the work products. Any deviation from the expected results will be documented.

Verification of the work products occurs when the work product is assessed against the inputs that were used to produce the work product. It is verified that the intent of the input work products is addressed. Test procedures are created to ensure this assessment is complete.

This activity also implements the analysis of the conformance of the work products to the established standards and procedures to assess the quality of the work products.

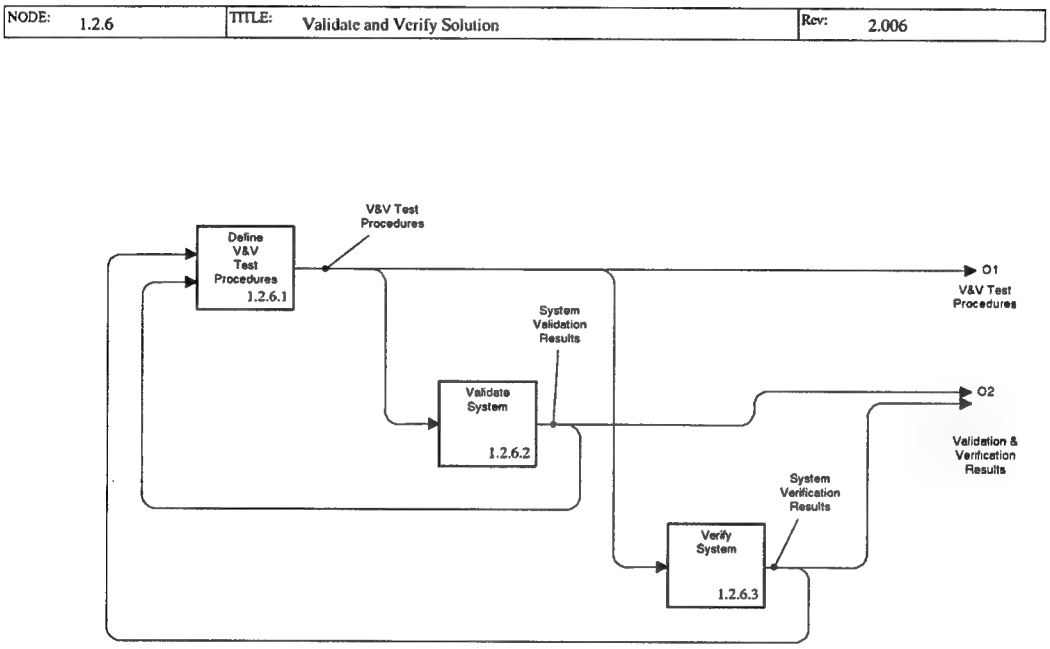


Figure 32. Validate and Verify Solution

The Define Validation and Verification Test Procedures activity (1.2.6.1) utilizes the work product definitions to define test procedures that verify and/or validate the work products.

The Validate System activity (1.2.6.2) assesses the work product to determine whether the customer/users' needs are being addressed. Test procedures are created to ensure this assessment is complete.

The Verify System activity (1.2.6.3) verifies the work product against the previous activities work products. Any activities output work products can be verified against the input work products used to create it.

3.4.2.6.1 Define Validation and Verification Test Procedures

The Define Validation and Verification Test Procedures activity (1.2.6.1) develops a clear definition of how the validation and verification is accomplished. In many cases, these procedures identify some form of V&V compliance matrix or are integrated into a system test plan. In any case, procedures are

defined so that an analysis is performed with an established set of criteria. These tests must ensure that work products conform to established quality measures. These procedures also document any quality assurance checks and balances that must be assessed.

3.4.2.6.2 Validate System

The Validate System activity (1.2.6.2) answers the question, "Are we developing the right system?" This activity ensures that the requirements for the system actually address the needs as stated by system stakeholders. This activity implements the validation test procedures and documents any variation between the expected results and the actual results. A root cause analysis is performed or initiated.

3.4.2.6.3 Verify System

The Verify System activity (1.2.6.3) ensures that the performance characteristics and constraints defined in all activities leading to the system definition are satisfied by the system. This activity implements the verification test procedures and documents any variation between the expected results and the actual results. A root cause analysis is performed or initiated. This activity also verifies that the defined systems engineering process is adhered to during the definition of the system.

3.4.2.7 Control Technical Baseline

Figure 33 contains the decomposition of the Control Technical Baseline activity (1.2.7). This activity maintains and versions the technical decision data that provides rationale for technical decisions that have been made. This activity also maintains the technical work products configuration control. This activity identifies the configuration items that constitute each historical baseline.

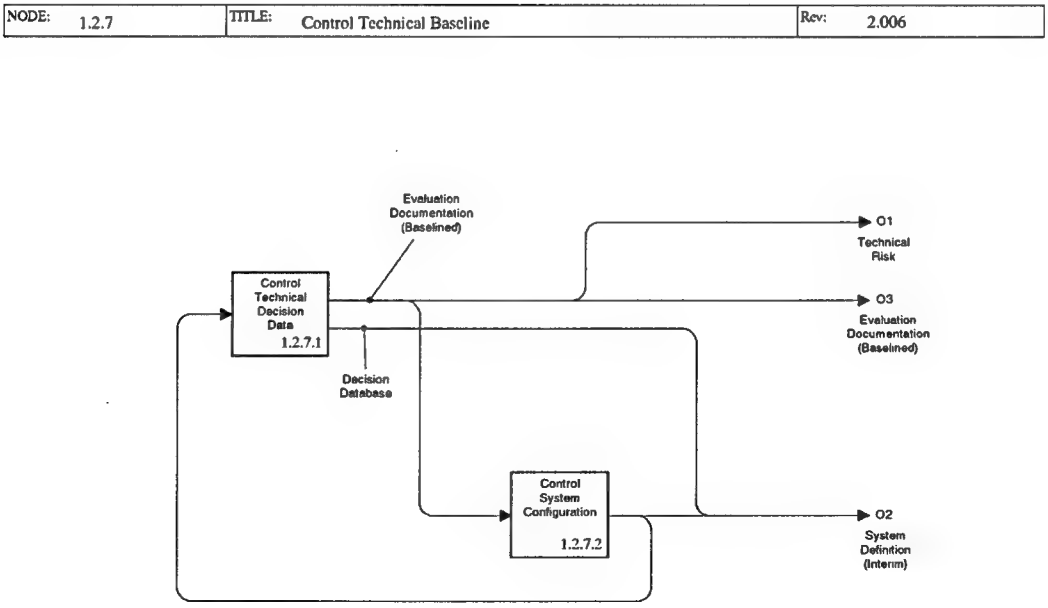


Figure 33. Control Technical Baseline

The Control Technical Decision Data activity (1.2.7.1) maintains the critical decision data that determined the selected system definition.

The Control System Configuration activity (1.2.7.2) maintains configuration control of all of the baselined system definitions. These system configurations may be intermediate ones that do not go to management.

3.4.2.7.1 Control Technical Decision Data

The Control Technical Decision Data activity (1.2.7.1) controls critical information pertaining to system definition decisions. This data is periodically baselined. This decision data serves as a library of key system development events and the key decisions made that have resulted in the current system. The data maintained by this activity is usually limited to information that may be needed by other system development activities, management, or reviewers. Data that is not critical to the program is usually maintained in a separate, inactive library.

3.4.2.7.2 Control System Configuration

The Control System Configuration activity (1.2.7.2) controls the baselined system definitions. As changes to the system are made at various levels, it is necessary to control a single baseline. This baseline contains information for all areas of system development. The baseline also documents the changes that distinguish the baseline from previous baselines. The baselining of information usually follows a well-defined configuration control process. In this process, proposed changes are reviewed and evaluated, the proposed updates are made, and the results of the baseline documented. The rigor at which this process is performed depends on the maturity of the system description. Early in a program, light control may be maintained to ensure that changes can be incorporated easily, but, as the system matures, the control of the configuration tightens in order to prevent changes that may have a ripple effect through the system. The level of control is also dependent on the level of system decomposition and the determination of the impact of any given change.

4. APPLYING TAILORING CONCEPTS TO GSEP

This section describes how to tailor the GSEP. It defines process tailoring terms and concepts, explains the process for tailoring the GSEP, and describes the varying degrees of detail that can be achieved for a process and how to create them by applying the tailoring process.

4.1 PROCESS TAILORING TERMS AND CONCEPTS

Sections 4.1.1 and 4.1.2 describe the process tailoring terms and concepts, providing an understanding of how GSEP defines process tailoring and how project characteristics are used as inputs into the tailoring process.

4.1.1 Process Tailoring

Process tailoring is creating a specific process from a general process. Process tailoring includes:

- ***Process Architecting.*** Process architecting is defining a set of processes and the interfaces between the processes. In the Automobile System example (Figure 5), process architecting defines the process(es) used to develop the Car (e.g., Manufacturing, Marketing, Distribution) and their interactions.

- ***Process Specification.*** Process specification is defining an activity by selecting what subsets of the inputs and outputs are mandatory, decomposing the activity into subactivities or tasks, and instantiating an activity to make it enactable. An activity is enactable when it is sufficiently detailed to allow it to be carried out by the resources assigned to complete it. Process specification is performing one or more of the following tasks:

  - **Selecting activity elements** includes deciding the activity's specific inputs and outputs. In the following discussion, the GSEP Analyze Risk activity (1.1.2) is used to illustrate the process specification concept described here. The Analyze Risk activity has a number of possible inputs (e.g., an approved EoS, user requirements, technology base, system implementation status, etc.) and produces one output, the approved RMP. Of those inputs and output, the EoS and the RMP are mandatory. (The entry and exit criteria define which inputs and outputs are mandatory.) During process specification each of the optional inputs and outputs for this particular activity are selected, and the specific available work products are assigned to the mandatory and selected optional inputs and outputs. For example, the assignments for the mandatory and selected inputs of this increment's Risk Analysis activity might be the EoS, the SOW, the specifications for the proposed systems and software engineering environments, and the current system schedule and costing figures. The process specification also determines what parts of the RMP are completed and output from this activity.

In addition to inputs and outputs, process specification also includes deciding what product and process measures to collect during the activity, how to verify the work performed during the activity, and what roles are to be associated with performing the activity.

- **Decomposing an activity** into subactivities and tasks specifies what work must be performed in the activity in more detail. In the Analyze Risk activity example, GSEP decomposes Analyze Risk into four subactivities: Perform Risk Analysis, Review Risk Analysis, Plan Risk Aversion, and Commit to Strategies. The Perform Risk Analysis activity might be decomposed into Identify Risks, Calculate Risk Factors, Evaluate Risks, Classify Risks, and Prioritize Risks; and Identify Risks might be decomposed into Analyze EoS, Analyze SOW, Analyze and Evaluate the Engineering Environment Specifications, and Analyze Cost and Schedule. The Analyze Estimate of the Situation activity might be decomposed into Identify Risks with Meeting Objectives, Identify Risks Associated with Constraints, and Identify Assumptions Risks, etc.

- **Instantiating an activity** is choosing the methods, tools, techniques, and resources for performing the activity. Process instantiation may also include deciding which standards apply to the activity.

  - **Choosing methods** determines how to accomplish the activity or part of the activity. For example, in the Analyze Risk activity, it might be decided to use a risk identification questionnaire to identify risks for this increment.

  - **Choosing tools** includes selecting the tool support for the activity. For example, if the risk identification questionnaire were automated, that might be one tool that is selected for performing the Analyze Risk activity.

  - **Choosing techniques** includes specifying an approach for performing a method(s) for an activity. For example, one technique for using the risk identification questionnaire is to have a number of development personnel complete the questionnaire then analyze the questionnaire results to identify risks.

  - **Assigning resources** entails determining what resources (e.g., people, equipment, facilities) are needed to perform an activity. For example, assigning Peter, Paul, and Mary to complete the risk identification questionnaires, reserving Conference Room A for the activity, and indicating that three copies of the questionnaire need to be supplied for the activity are all part of assigning resources in the Analyze Risk activity.

  - **Selecting standards** includes determining what specific standards are to be followed during the activity. The standards can be customer standards, government standards, corporate or project standards, or general systems engineering standards. For example, deciding to use the Organizational Risk Identification Questionnaire, should one exist, for the risk identification is part of selecting standards. If the organizational risk identification questionnaire needed to be slightly modified before it could be used, then deciding what modifications were needed and creating the modified risk identification questionnaire would also be part of process instantiation.

4.1.2 Process Drivers

Process tailoring occurs in a particular context (goals and environment), and this context impacts the process tailoring and the resulting tailored process. The elements in the context that impact the resulting process are called **process drivers.**

The process drivers include objectives (or goals), constraints, alternatives, and risks that are considered during process architecting and process specification. Process drivers fall into three categories:

- ***Technical.*** Technical process drivers are specific characteristics of the system being developed. Some examples of technical process drivers are the need for rapid prototyping to solidify user requirements, the need for an early concept design demonstration to get customer buy-in and commitment, and the need to evaluate emerging technologies for use on the development effort.

- ***Organizational.*** Organizational process drivers are specific characteristics of the organization performing the work (e.g., structure, economic, political, personnel). Some examples of organizational process drivers are the need for low-budget alternatives, the need for skills training, and the need to select and manage a subcontractor.

- ***Process Constraint.*** Process constraint process drivers are characteristics of the development process that constrain the tailoring process. Some examples of process constraint process drivers are the constraint that productivity metrics must be collected, the constraint for customer reviews at the end of every increment, the constraint that measurable success criteria must be defined for each increment, and the constraint that one of those measures must be how well the success criteria are met.

4.1.2.1 Drivers for Process Architecting

During process architecting, process drivers are used to determine the most suitable general processes for the system and to determine the needed process interfaces. In the Automobile System example (Figure 5), process drivers are used to influence the selection of the process that will be used to develop the Car Subsystem, Manufacturing Subsystem, or Marketing Subsystem. The processes used to develop each system can be different. It is possible, and reasonable, to have different processes for different subsystems because the selection of the most appropriate process for a particular subsystem is based on its unique process drivers. Examples of process drivers that impact general process selection are:

- ***Process Flexibility.*** Does the selected process need a lot of process tailoring? How flexible is each process and how easily can it be tailored to the particular needs of the system?

- ***Process Familiarity.*** How experienced is the staff at following the selected process and how much process training is needed?

- ***Process Suitability.*** How well is the selected process suited for meeting the objectives of the particular system?

During process architecting, process drivers are also used to determine the necessary process interfaces. For example, in the Automobile System there may be a need to build an inexpensive car

(an organizational process driver). That process driver might result in the need to have interfaces between the Manufacturing, Car, and Marketing Subsystems. One purpose of these interfaces would be to ensure that the car and manufacturing subsystems developed an inexpensive car, yet did not eliminate any features that made the car less marketable. These types of trade-offs are only possible if the communication and decision-making interfaces are established during process architecting.

During process architecting, after the process(es) have been selected, it is possible to identify the drivers that constrain the architecture. These drivers are used during the remainder of process architecting to define the necessary process interfaces. They are also used during process specification when the activity inputs and outputs are being selected and during activity decomposition. Examples of process drivers that are based on the selected process(es) and that impact process architecting and specification are:

- ***Required Activities.*** The process(es) selected may require activities such as periodic customer reviews, risk management activities, and migration plan creation and update.
- ***Required Activity Elements.*** The process(es) may require that all activity definitions contain specific elements such as collection of specific measures, verification of all activity tasks, and indication of the primary and secondary roles responsible for performing an activity.
- ***Required Organizational Procedures.*** The development organization may have procedures that all processes must follow. Examples of these procedures are the requirement for a senior staff member to have final approval on all work products before the activity is considered complete, the need for following organizational style guidelines when creating work products, and the need to have a complete cost accounting of all resources used during the performance of an activity.

4.1.2.2 Drivers for Process Specification

Process drivers are also used during process specification. Process drivers influence what **optional** activity inputs and outputs are included and what available work products are specified for each input and output. Process drivers also influence the activity decomposition and instantiation. Examples of process drivers for process specification are:

- ***Available Inputs and Mandatory Outputs.*** Process specification must specify which of the high-level inputs and outputs defined in the general process should be associated with each specific activity.
- ***Needed Guidance.*** The level of detail that is required for each activity definition is dependent on the level of experience of the staff members who will be performing the activity.
- ***Number of Customer Reviews.*** The number of reviews with the customer and how far apart they are spaced is dependent on the rapport that has been established with the customer and how much information exchange and buy-in is necessary to ensure successful customer relations.
- ***Required Standards.*** The required standards impact the methods used to perform the activities and the outputs that the activities produce.
- ***Available Tools.*** Availability of tools and equipment impacts the decision of what methods to use and which activities to automate.

- ***Training Requirements.*** The need for special skills requires the time and resources to provide training.

Although process architecting precedes process specification, they are not done independently. They share many of the same process drivers, and a coordinated effort to create the process must take place. Like most engineering, there is looping and iteration between the architecting and specification activities that ensure the creation of a usable process. The tailoring process defines how the process architecting and specification are done and how the process drivers are used in process creation.

4.2 TAILORING THE GSEP

In GSEP, process tailoring is important because it creates the specific systems engineering processes that are enacted on a project. A process architecture defines the way that the instantiated GSEPs work together to define the "systems engineering process" for the whole system. For example, Figure 34 shows a subset of a sample process architecture for the Automobile System. In this example, there are interactions between the Drive Shaft and the Drive Shaft Pressing components. Specifically, in this example, these interactions are the exchange of information contained in the System Definition, System Plan, and System Status documents for each component.

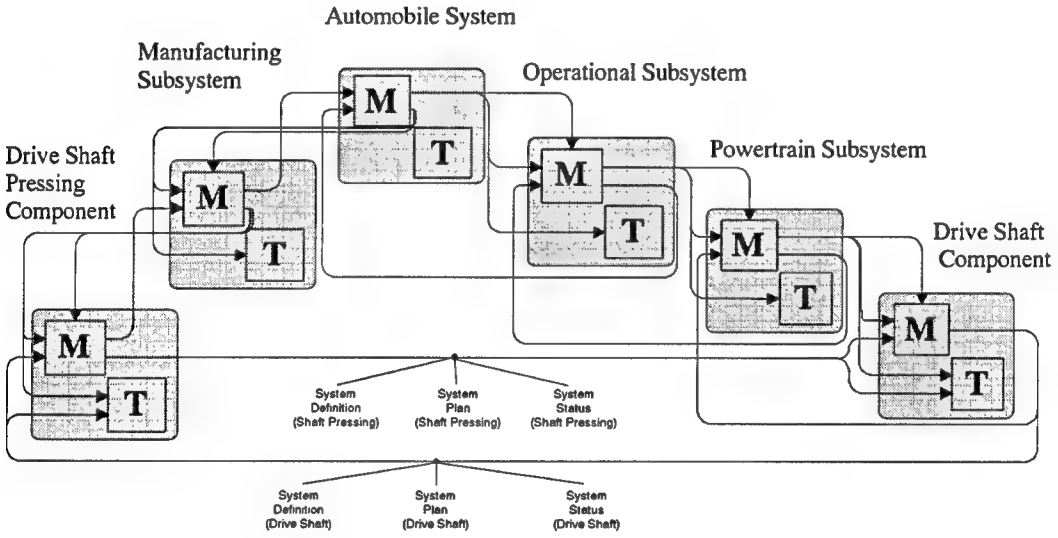


Figure 34. Sample Process Architecture for the Automobile System

The following list describes how the remainder of Section 4 addresses the creation of processes that will be used to engineer the system architecture:

- An engineer can use the GSEP to define several different types of processes. Section 4.2.1 describes the three types of processes that may be created from the GSEP.

- The GSEP can be used to create the systems engineering process architecture. Section 4.2.2 describes the creation of the process architecture and shows how process tailoring is used to define the processes for creation of the main system and its subsystems and components.

4.2.1 GSEP Process Types

The following three types of processes may be created from the GSEP and are necessary for implementing the GSEP on a system development effort:

- ***The System Plan Process.*** This type of process creates a high-level system plan for the entire system. This plan is used to initiate the tailoring process and the systems engineering process for the system (see Section 4.2.1.1).

- ***The Tailoring Process.*** This type of process creates tailored systems engineering processes (see next bullet) and is used for two purposes (see Section 4.2.1.2).

  - ***Initial Process Generation.*** During initial process generation, the GSEP is tailored to create an enactable process.

  - ***Continuous Process Improvement.*** During continuous process improvement, the process that was created during initial process generation is modified based on lessons learned from implementing the process.

- ***The Systems Engineering Process.*** This type of process creates system parts. It is what is traditionally referred to as the systems engineering process (see Section 4.2.1.3).

4.2.1.1 System Plan Process

All systems engineering efforts begin with the creation of a high-level plan. Because the quality of this plan will impact the entire development, the plan's creation should follow a well-defined process. In the GSEP, the System Plan Process is the process that is used to create the initial high-level plan for the engineering effort. In this process, the management part of the GSEP is used to schedule and track the development of the system plan, and the technical activities are used to create the system plan itself.

4.2.1.2 Tailoring Process

Creating a Tailoring Process. The systems engineering process itself can be viewed as a system. As such, the GSEP can be used to engineer the systems engineering process. The tailoring process uses the GSEP definition and process drivers to produce the systems engineering process that will be used to develop the system. The System Tailoring Process (see Figure 35) is a tailoring process that defines the Main Systems Engineering Process.

The GSEP must be instantiated to be used to define the System Tailoring Process. This instantiation defines the tools, methods, and procedures that are used to create and maintain the System Tailoring Process. In terms of the GSEP itself, the only thing that changes is the context. The customer potentially includes the customer purchasing the system being engineered by the Main Systems Engineering Process, the systems engineering organization that is producing the main system, and/or the organizations producing the components or subsystems that make up the system. The users include all parties that are responsible for producing or modifying the system. The GSEP is one of the methods used to define the System Tailoring Process. The GSEP is also viewed as an external system definition that defines the requirements for the Main Systems Engineering Process.

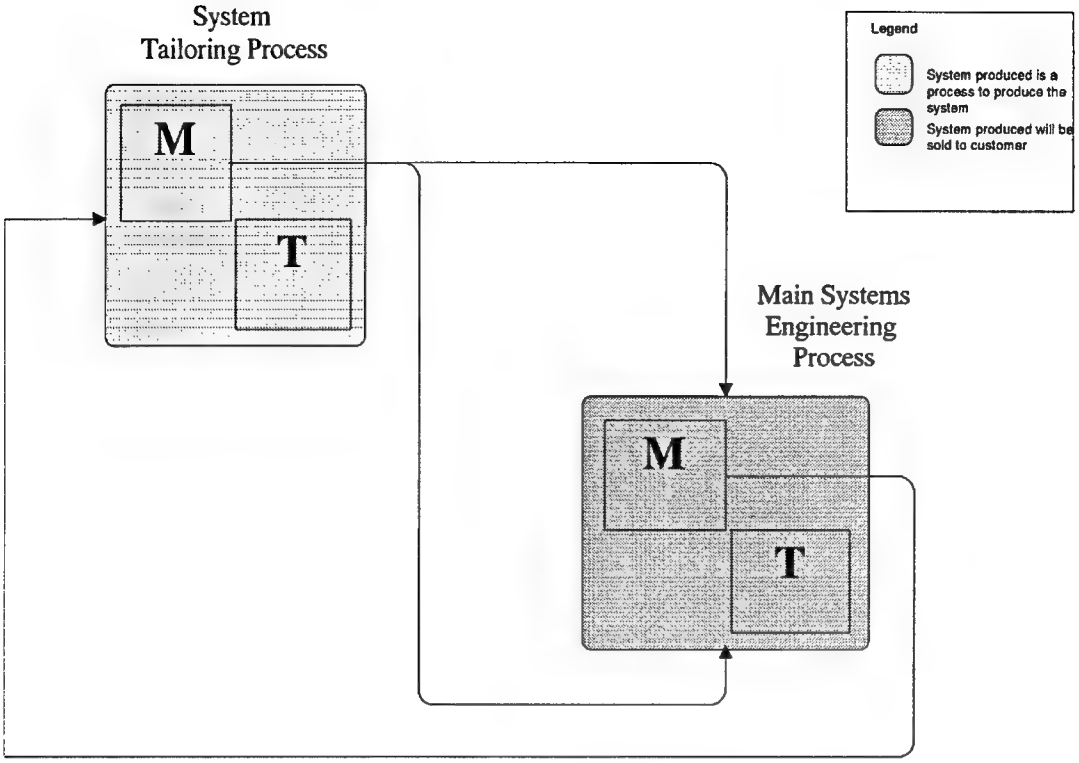


Figure 35. Tailoring Process for Systems Engineering

Dealing With System Decomposition. One of the means of dealing with complexity is the decomposition of the system (see Section 3.3.2). This decomposition applies just as well to the tailoring processes as it does to the system being built. The processes created to support the definition of subsystems must also interact in well-defined ways that meet behavioral and performance requirements that are defined during the design of the higher level system concept.

The System Tailoring Process defines the process for the system being built (see Figure 35) and defines the process tailoring architecture (see Figure 36). The process tailoring architecture defines the interactions between:

- ***Subsystem and/or component tailoring processes.*** These are interactions that state things such as "these two subsystems must pass system definitions back and forth" or "metrics for performance of subsystem A must be communicated to subsystem B."

- ***Subsystems and/or components and the System Tailoring Process.*** These are interactions that state things such as "the subsystem must be able to provide quality and performance measures in form Y to the system" and "the system must provide the training for method Y that is needed for subsystem X."

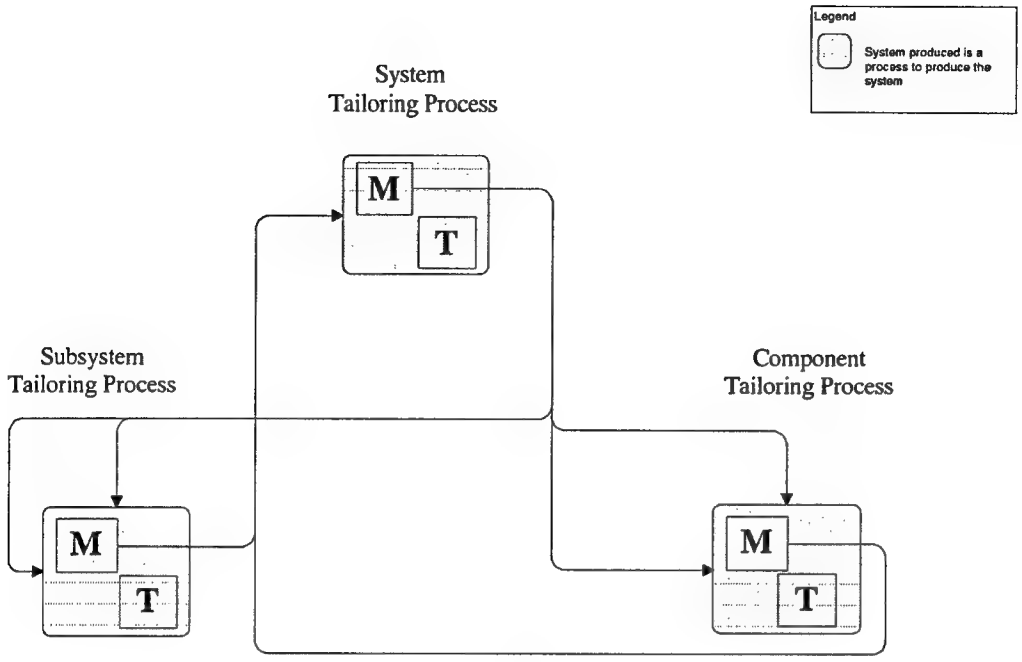


Figure 36. Process Tailoring Architecture

Establishing the Tailoring Process Interfaces. The tailoring process creates and continuously improves the systems engineering processes. It must interface with the tailoring processes of other systems being engineered, with organizational process definition and improvement efforts, and with the systems engineering processes it created for an engineering effort. Consequently, its context is larger than just the systems engineering processes it creates.

In this larger context, the tailoring process users also include all parties that are responsible for producing or modifying subsystem or component tailoring processes (see Figure 36) and other users that are responsible for process definition and improvement work in the organization. The external system definition and subsystem/component status come either from this system's lower level tailoring processes (if any subsystems or components exist) or from the tailoring processes of other systems. The external system plan and the higher level external system definition, which provide many of the technical requirements for the process, come from the higher level systems engineering process and the System Tailoring Process.

Continuous Process Improvement. Continuous process improvement occurs as a result of the information flow back up into the System Tailoring Process from the Main Systems Engineering Process (see Figure 35) and from the lower level Subsystem/Component Tailoring Processes (see Figure 36). The information that flows back up into the System Tailoring Process is used to identify and then address issues that are discovered in the Main Systems Engineering Process. These issues are then addressed by the the System Tailoring Process, resulting in updates to the Main Systems Engineering Process.

4.2.1.3 Systems Engineering Process

The systems engineering process defines the process that takes the needs derived from the customer and user inputs and produces the system that meets those needs (e.g., the Automobile System in Figure 5).

4.2.2 Defining a Systems Engineering Process Architecture

The GSEP can be used to derive the systems engineering process architecture. The preferred approach to create this architecture is a recursive one and is defined in the same way that the system architecture for the actual system is defined (which is also done recursively). Figure 37 shows a sample process architecture. The step numbers in the figure correspond to the steps used to define a systems engineering process. The list following the figure describes these steps.

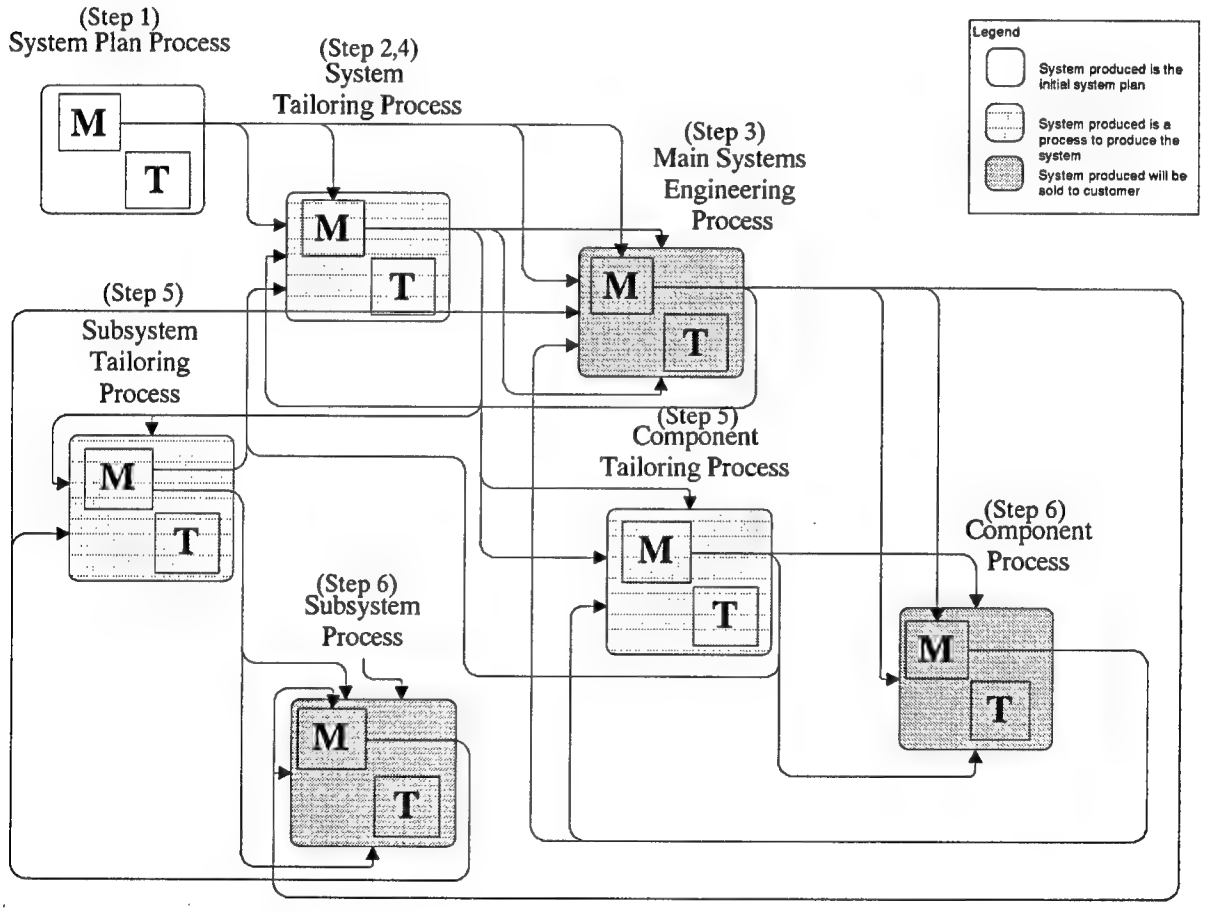


Figure 37. Sample Process Architecture

The following steps are used to define a systems engineering process equivalent to the sample process architecture from the GSEP:

Step 1. The System Plan Process is used to create a System Plan.

Step 2. A System Tailoring Process is created based on system requirements documented in the System Plan.

Step 3. The Main Systems Engineering Process is created using the System Tailoring Process and produces the system architecture.

Step 4. The system architecture is used by the System Tailoring Process to define the Subsystem/Component Tailoring Processes and the resultant process tailoring architecture.

Step 5. The Subsystem/Component Processes are created using the Subsystem/Component Tailoring Processes.

Step 6. Repeat Steps 3 through 5, viewing each Subsystem/Component Process as a Main Systems Engineering Process.

Note that Step 1 and 2 do not repeat.

4.2.2.1 Using Process Tailoring to Define the Main System

To understand the discussion in Sections 4.2.2.1 and 4.2.2.2, the reader must refer to the context for the GSEP that is shown in Figure 12, Section 3.2. In this figure, the context for the GSEP includes:

- ***Inputs.*** Subsystem/Component Status, External System Definition, Customer Requirements, User Requirements, and the Technology Base
- ***Outputs.*** System Plan, System Status, System Definition
- ***Constraints***. External System Plan and Organization Plan
- ***Mechanisms (Tunneled)***. Resources, Methods/Algorithms, and Tools

The GSEP was designed so that it can be used, through tailoring, to define each of the various types of processes without modification, by mapping these contextual inputs, outputs, constraints, and mechanisms to the appropriate context.

The discussions that follow describe how to create the Main Systems Engineering Process and the Subsystem and Component Processes. Each discussion describes how to define the GSEP context so that it can be tailored to the appropriate process.

To create the Main Systems Engineering Process, two processes must be created first: the System Plan Process and the System Tailoring Process.

The System Plan Process. The inputs to the System Plan Process are those customer, user, and corporate inputs necessary to outline a top-level System Plan that calls out the overall strategy for building the system. The System Plan Process is different from the standard GSEP process described in Section 3 in the following ways:

- There are no subsystem or components that give the subsystem/component status.
- There is no external System Plan or external System Definition.
- The increments in the management part of the process are the increments necessary to create an appropriate System Plan.

- The requirements in the technical part of the process are the requirements for the System Plan (e.g., when the plan must be finished, what format or standard template the plan must follow, and the level of detail contained in the plan).

The System Tailoring Process. In the context for the System Tailoring Process, the customer potentially includes both the customer purchasing the system and the systems engineering organization that is producing the main system architecture. The users include all parties that are responsible for producing or modifying the system architecture for the main system. The external system definition and subsystem/component status come from the Subsystem Tailoring Processes and the Component Tailoring Processes. The external system plan comes from the System Plan Process.

In creating the Main Systems Engineering Process, the System Tailoring Process defines any lower level activities, as well as instantiating the activities by assigning the methods, tools, techniques, etc. that are used to carry out the definition of the main system.

The System Tailoring Process defines life-cycle process architecture for the system. The life-cycle process architecture defines the first-level decomposition of the system that includes the Operational, Manufacturing, Support, and other subsystems.

Defining Life-Cycle Subsystems. The Main Systems Engineering Process creates the life-cycle architecture that defines the life-cycle subsystems and allocates top-level system requirements to these subsystems. The Main Systems Engineering Process defines an architecture that includes all of the interfaces between and requirements for the subsystems (e.g., Manufacturing, Operational, Marketing Subsystem). This systems architecture is included in the system definition for the main system as part of the system design. This system design provides the requirements for the life-cycle subsystems.

Process Interfaces and Process Improvement. The life-cycle process architecture defines the process interfaces for the Operational, Manufacturing, Support, etc. Subsystems. It also ensures that there is continual process improvement and that the appropriate information is shared between the life-cycle processes. In the sample architecture in Figure 37, the process architecture defines the interactions between the Subsystem Tailoring Process and the Component Tailoring Process. If the subsystem is decomposed into multiple components, the process architecting for those components is done in the Subsystem Tailoring Process, not in the System Tailoring Process.

The System Tailoring Process also defines the interactions between its Subsystem and Component Processes and the Main Systems Engineering Process (see Figure 37). This includes the flow of the system plan and system definition down to the subsystems and components, and the subsystem/component definition, status, and plan back up to the Main Systems Engineering Process.

4.2.2.2 Using Process Tailoring to Define Subsystems and Components

The Subsystem/Component Tailoring Processes create the Subsystem/Component Processes as well as the subsystem/component process architecture. Again, the GSEP can be tailored to create these processes by changing its context. The customer potentially includes the customer purchasing the subsystem/component, the systems engineering organization that is producing the main subsystem/component architecture, and/or the organization producing the system of which the component or subsystem is a part. The users include all parties that are responsible for producing or modifying the subsystem/component architecture. The external system definition and subsystem/component status

come from lower level tailoring processes (if any exist) and any other tailoring processes that were defined in the process architecture by the System Tailoring Process. The external system plan and the higher level external system definition that provide many of the technical requirements come from the Main Systems Engineering Process and the System Tailoring Process.

In creating the Subsystem/Component Processes, the tailoring process completes the process specification for the Subsystem/Component Processes. This includes definition of any lower level activities, as well as instantiating the activities by assigning the methods, tools, or techniques, that define the subsystem/component.

The Subsystem/Component Process creates an architecture that defines subsystems, components, and/or elements. This Subsystem/Component Process defines an architecture that encompasses all of the requirements for and interfaces between its constituent subsystems, components, and elements.

The subsystem/component process architecture defines how any constituent subsystems and components will work together to ensure that there is continual process improvement across the set of processes under consideration and that the appropriate information is shared between the constituent subsystem and component processes. The process architecture also defines any interactions between the constituent subsystem/component process tailoring activities.

4.3 CONCLUSIONS

The GSEP tailoring process is used to create all the other GSEP processes. Through **process architecting** and **process specification,** an infinite number of variations can be derived from the GSEP. In each of the processes that are created through **process tailoring**, the fundamental GSEP concepts remain constant. This is because process tailoring only includes process architecting and specification. Process tailoring does not include changing the process. For example, process tailoring would never include eliminating an activity, adding an activity that was not a refinement of an existing activity, adding or deleting mandatory inputs or outputs, or modifying entry or exit criteria. If this type of change is made, then the resulting process is not the same as the original process from which it was created. Process changes like this are part of process engineering, that is, creation of a new process from an existing process(es).

Typically, an organization uses the GSEP to create an **organizational systems engineering process.** The organizational process may be a tailoring of the GSEP, or it may be a reengineered process that is based on the GSEP. The GSEP was designed to be generic and, as such, an organization should be able to address all their systems engineering concerns within its framework. If, during an attempt to tailor the GSEP, an organization finds the GSEP requires modification, care should be taken to ensure the fundamental GSEP concepts (see Section 2 and 3) are not violated. In addition, organizations that reengineer the GSEP should be cautious and consider the following factors before deciding that reengineering the GSEP is best for their organization:

- ***Process Tailorability.*** The GSEP was designed to be tailorable. Thus, it is possible to tailor the GSEP for applications other than systems engineering, such as the creation of a tailoring process and the creation of a process to create a System Plan. The GSEP is also tailorable to different systems engineering domains. If GSEP is reengineered so that it does not address the key systems engineering concepts, it would be less appropriate for use as an organizational process.

- ***Process Enactability.*** The GSEP was created so that it could easily be made enactable. If the process that is created is difficult to enact, then the burden of enacting the process is levied on the practitioners of the process. This can be a heavy burden because most practitioners are not process engineers.

- ***Process Improvability.*** The GSEP provides the means for process improvement because it recognizes that processes need to be continually improved. It is not necessary that the process improvement activities be part of the general process. The GSEP separates the process creation and improvement from the system creation. If process improvement is embedded in the process that is used to develop the system, then that development process becomes more complex and process concerns are not separated from development concerns. (See the Separation of Concerns discussion in Section 2.3.4.) Regardless of the approach used, the organizational process must have a mechanism to ensure that continuous process improvement is done.

The organizational process(es) is the basis for all other processes in the organization. If the organizational process is to generate processes as needed, a suitable tailoring process is required. The GSEP tailoring process can be used, or another tailoring process must be selected or created.

This page intentionally left blank.

5. GSEP-BASED ENGINEERING FOR PRODUCT FAMILIES

This section discusses a GSEP-based approach to systems engineering for product families. A **product family** is a formalization of a product line, which is a collection of similar existing and potential systems that address a designated business area market. Systems engineering for a product family comprises creating and maintaining both a product family and a capability to produce and deliver specific members of the product family to customers. Section 2.4 describes the concept of product families in more detail.

The Consortium approach to reuse-driven software engineering, known as the Synthesis methodology, is to construct each system as an instance of a family of systems that have similar descriptions (Campbell, Faulk, and Weiss 1990). The *Reuse-Driven Software Processes Guidebook* (Software Productivity Consortium 1993a) describes the concepts and practice of software engineering to create and use product families. This section describes the extension of Synthesis to GSEP-based systems engineering for product families.

5.1 RATIONALE FOR ENGINEERING WITH PRODUCT FAMILIES

An organization's motivation for developing a product family is that it recognizes a viable market for a series of similar systems. It must have both sufficient expertise to develop other systems of the same type, generally based on prior experience in developing or maintaining such systems, and a business commitment to do so. A customer needing a critical system that is expected to evolve to meet changing needs over many years suffices as such a market.

A conventional approach to systems engineering emphasizes one-of-a-kind handcrafting of each system. Although previous systems' components are sometimes used in developing a system, such components are seldom designed with sufficient attention to the needs of future systems and, therefore, often require ad hoc modification to meet those needs. Such an approach is inefficient for developing a series of similar systems. In comparison, a capital investment in the creation of a **domain,** comprising a product family and an associated capability to produce deliverable systems, can produce substantial long-term benefit.

A domain standardizes an organization's understanding of a set of similar problems and their corresponding system solutions. The essence of engineering for a product family is identifying and exploiting similarities in the systems that an organization builds to avoid redundant work. Each system in a domain may be large and complex, such as a space station; moderate, such as an automobile; or small, such as the engine of an automobile. Similarly, a domain may be large, encompassing substantial variety in the included systems, or small, encompassing only a small number of alternative systems having limited variety.

Using a domain to create deliverable systems can yield the following benefits:

- Reduced time and cost to develop a system (including the amortized cost of creating the domain)
- Higher quality systems, with less effort to correct and reverify flaws detected in verifying work products, through repeated flexible reuse of standardized parts
- More emphasis on understanding customer needs specific to each system, expressed in a standardized form, and a capability to identify and resolve unclear or incomplete requirements based on an explicit set of supported product variabilities
- Less dependence on direct, dedicated participation in each project by key engineers whose knowledge and expertise in problems and their solutions make a domain feasible
- An ability to rapidly produce alternative systems (or, equivalently, system models) for evaluating trade-offs in satisfying customer needs
- Increased standardization, across a product line, of work products which are consistent and integrated by design rather than by brute force
- Improved communication and coordination among engineers based on a shared understanding of how systems to be produced are alike and how they can differ
- A sounder maintenance capability that ensures the continuing consistency and integrity of all work products of a system
- Better support for bid-and-proposal efforts based on a more accurate understanding of the organization's existing capabilities to deliver a particular type of system

Engineering for product families is most viable in an organization that emphasizes long-term business objectives. There must be a perceived market for a family of similar products, such as multiple customers with differing needs, a single customer who needs multiple versions of a system, or a long-lived system that is likely to undergo significant changes in customer needs or technology. Although creating a product family can take an investment of time and money that is greater than that required to produce a single system, it will reduce development and maintenance costs across an entire product line.

5.2 A PROCESS TO CREATE AND USE PRODUCT FAMILIES

The process of engineering for a product family must be designed to support two independent but interrelated objectives:

- To produce and deliver systems
- To increase the effectiveness (profitability, productivity, product quality, manageability, and responsiveness) of system production and delivery

To address these separate concerns most effectively, the process consists of two integrated subprocesses: **application engineering** and **domain engineering**. Figure 38 shows how these processes relate to each other.

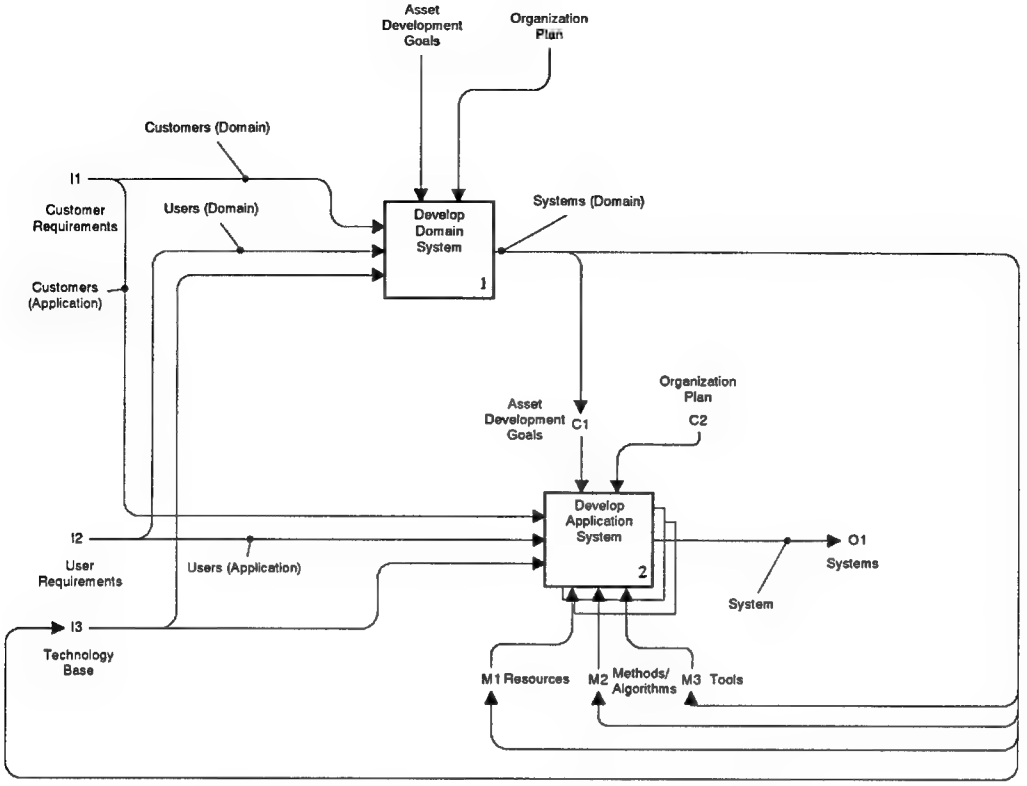


Figure 38. A Process for Creating and Using a Product Family

Application engineering (or Develop Application System) is the process followed to produce and deliver systems. Domain engineering (or Develop Domain System) is the process followed to increase the effectiveness of system production and delivery. Figure 39 is derived from Figure 11 which depicts the process used to independently develop several systems by an organization. Figure 39 organizes this process to develop and use a product family. Domain engineering supports application engineering by supplying projects with a domain consisting of a product family and an associated systems engineering process and environment tailored to the projects' needs. (The Above, Across, and Below process relationships between any system and other systems are omitted only to simplify this figure. The discussion concerning Figure 11 provides more on this.) An important aspect of the relationship, which is not shown in the figure, is feedback from application engineering projects to domain engineering that supports further evolution of the domain to better meet future project and customer needs.

Application engineering is a standardized process by which projects produce and deliver application systems to customers. In terms of objectives, this is the equivalent of conventional, one-of-a-kind systems engineering. Both conventional and product-family approaches start with customer requirements and produce a set of work products that are meant to satisfy those requirements. Either approach may use COTS assets or reuse assets produced in developing previous systems. However, in a product family approach, through domain engineering, an organization institutes a simpler, more efficient process of automated system development, supported with standardized, reusable work products.

Application engineering creates and validates a model that is used to generate a required application. The application engineering process focuses effort on requirements and engineering decisions that are sufficient to describe a particular system, given a family of such systems. Work products, including hardware (such as instructions to a silicon compiler, a programmable device, or a circuit board layout assembler), software (including both deliverable and support programs), procedures (such as for use, for testing, or for installation of a system), and documentation (such as the GSEP's system definition, design documents, or user manuals), are derived from these decisions using adaptable forms of those work products provided by domain engineering.

Domain engineering supports application engineering in two ways. First, it creates a set of adaptable work products that correspond to the work products that application engineering projects must produce. Domain engineering creates work products that are adaptable to subsequent application engineering decisions by identifying key decisions that are deferred until a particular system is needed and parameterizing each work product to show how it varies as a result of those decisions. Second, domain engineering defines a standard application engineering process that supports the decision making and the work-product creation appropriate to projects in the business area. The process definition institutes standard procedures and practices that domain engineering augments with an appropriate automated support environment.

5.3 USING THE GSEP IN ENGINEERING FOR PRODUCT FAMILIES

The *Reuse-Driven Software Processes Guidebook* provides basic definitions of domain and application engineering processes. To support systems engineering, these definitions must be tailored in the context of GSEP. However, having been developed before GSEP, these processes as presented here deviate somewhat from pure GSEP-derived processes. Specifically, the scope of the domain engineering process definition is beyond that of GSEP because it includes activities of the Manufacturing, Deployment, and Operation and Support phases of the "domain system" life cycle. Examples of this are activities that identify the steps to implement an application engineering environment or to support a domain's users during and after delivery. In addition, the application engineering process definition is only intended to be suggestive of what domain engineering should define as an application engineering process.

Figure 39 is a snapshot of the relationships between GSEP-based domain and application engineering processes. Note that a domain may correspond to a family of similar systems, subsystems, or components. A GSEP-based domain engineering process includes the engineering of both a product family as reusable assets that are used to build applications and a tailored, GSEP-based application engineering process. The product family is provided as a part of the technology base input to the application engineering process. A GSEP-based instantiation of the application engineering process includes an environment comprising tools, methods, and resources that are used to build applications (i.e., family members). Note that the product family developed in domain engineering is provided to all application engineering projects that follow the associated GSEP-based application engineering process.

5.3.1 GSEP-Based Domain Engineering

The scope of domain engineering is a domain, corresponding to an organization's total 'systems engineering system.' A domain includes all application systems to be created over time for customers within a targeted market, as well as a process and supporting environment for developing and

delivering those systems. The leveraged Synthesis domain engineering process defined in the *Reuse-Driven Software Processes Guidebook* consists of five activities which correspond, as follows, to GSEP activities:

- ***Domain management***, which is equivalent to GSEP's Manage Development Effort activity for a domain
- ***Domain definition***, which is equivalent to GSEP's Analyze Needs and Define Requirements activities for a domain
- ***Product family engineering***, which is equivalent to a GSEP process for the "domain system" subsystem corresponding to the product family component of a domain
- ***Process engineering***, which is equivalent to a GSEP process for the "domain system" subsystem corresponding to the process /environment component of a domain
- ***Project support***, which is equivalent to GSEP's Validate and Verify Solution activity for a domain, but also includes installation, delivery, and support

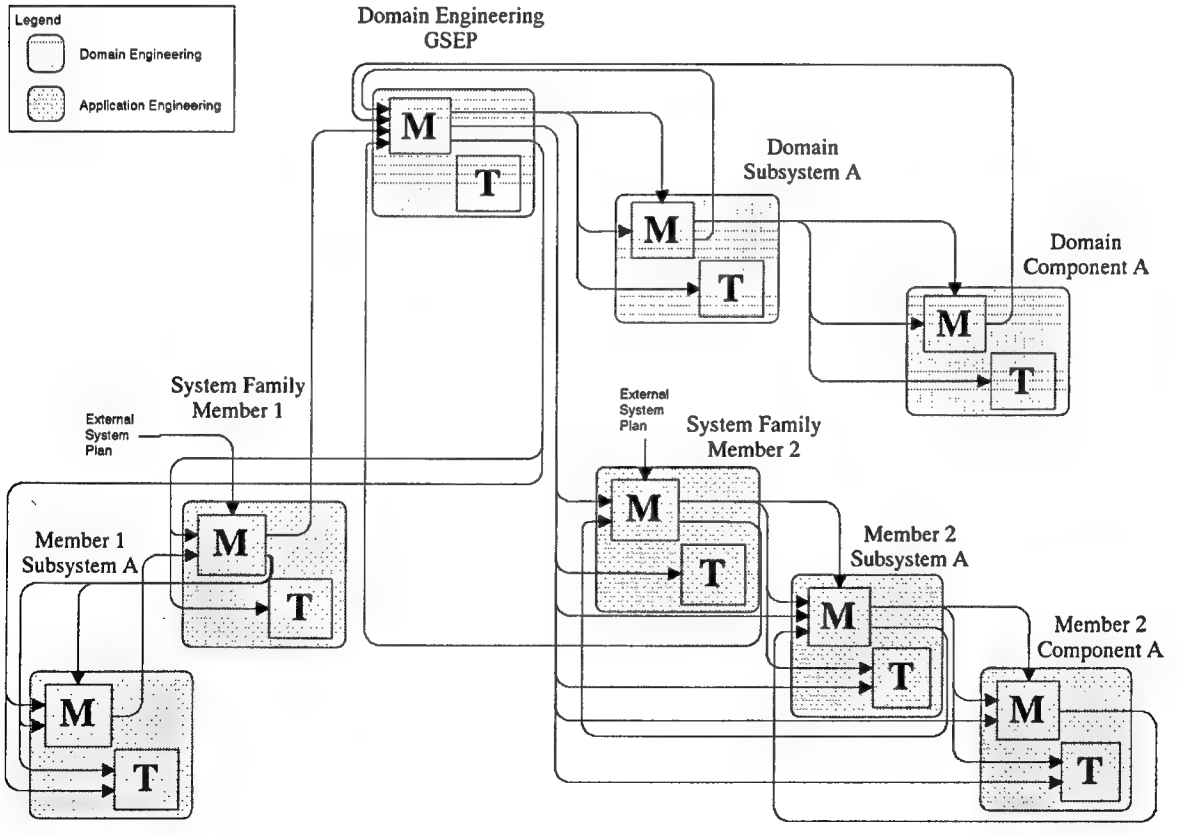


Figure 39. Interactions Between GSEP-Based Domain and Application Engineering Processes

GSEP activities remain much the same within a domain engineering process, except that each is oriented to a family of systems characterized by commonalities and variabilities rather than to a single system and, therefore, requires creating adaptable work products. For this reason, several GSEP process terms have special interpretations within the domain engineering context:

- *System.* A domain, corresponding to a family of application systems, subsystems, or components and comprising the product family and the associated application engineering process and environment.
- *Customer Requirements.* Business objectives, partially based on the known and projected needs of the market (customers) for products.
- *User Requirements.* Domain (problem aspects) knowledge, held by managers and engineers responsible for the application engineering of systems represented by a domain and by users of those systems.
- *Technology Base.* Domain (problem aspects) knowledge held by managers and engineers responsible for the application engineering of systems represented by a domain, including legacy products that are representative of the domain.

Product family engineering creates a product family, represented by requirements, design, and implementation work products, that is sufficiently adaptable to represent any member of the product family. This includes an adaptable architecture and assets that are flexibly reused to generate product family members. Rather than being a direct source of revenue, a domain is an investment, primarily intended as a means to leverage the product development efforts of the organization for greater profitability on each product, product version, or product upgrade.

The domain engineering process can be followed to create a domain at the system, subsystem, or component level. Within a domain, each subsystem or component may be an integral part of the domain, a member of a family forming a distinct domain, or a conventionally developed system.

5.3.2 GSEP-Based Application Engineering

Application engineering produces a system that is a product family member and that meets the particular needs of a customer and users. Domain engineering defines a product family and a tailored application engineering process to make this a viable capability. This section describes one form that the process could take, based on the definition of application engineering for the leveraged Synthesis process in the *Reuse-Driven Software Processes Guidebook,* but tailored based on GSEP. This process, as presented here, assumes that domain engineering is tasked to minimize application engineering effort by providing projects with a comprehensive and consistent set of reusable assets and maximum automated support. More modest versions of the process would leave application engineering projects responsible for more of the effort as one way to limit the needed investment in domain engineering.

Domain engineering is able to create a more effective application engineering process by standardizing the requirements, design, and implementation of systems in the domain. Standardization means that work related to similarities among encompassed systems can be completed once by domain engineering. This leads to standardized work products that can then be generated in a tailored form from domain-specific reusable assets. Because standardization provides a foundation for

increased automation, the application engineering effort tends to be condensed into a decision making process emphasizing problem/solution modeling and analyses of alternatives. Each application engineering project focuses its efforts on resolving issues that are specific to the requirements and constraints motivating a particular system.

The leveraged Synthesis application engineering process consists of four activities that correspond, as follows, to GSEP activities:

- ***Project management***, which is equivalent to the GSEP Manage Development Effort activity
- ***Application modeling***, which is equivalent to the GSEP Define System activity
- ***Application production***, which may be fully automated but is outside the scope of GSEP in any case
- ***Delivery and operation support***, which is outside the scope of GSEP

Systems produced by application engineering are usually intended to bring in revenue to an organization. Domain engineering defines the activities and work products of the application engineering process to be used by projects so that they have the means to be more profitable. Feedback from application engineering projects to domain engineering is used to improve the domain as a 'systems engineering system.'

The inputs to application engineering from domain engineering include an extended technology base that includes work products defining the product family (as a set of reusable assets), methods and tools in an environment that supports the activities of the application engineering process, and, possibly, resources to perform application engineering activities.

5.3.2.1 Tailoring Management Activities for Application Engineering

Domain engineering tailors the following two GSEP management activities for application engineering, in order to simplify them within the context of a domain (see Figure 40):

- ***Understand Context Activity (1.1.1).*** Domain engineering provides application engineering with an ability to generate an EoS when supplied with specified information about context for the product; risks and assumptions that have been inherited; and organizational, political, economic, and technical issues that have been identified.
- ***Analyze Risk Activity (1.1.2).*** Domain engineering simplifies the Analyze Risk activity by constraining and formalizing the potential risks that are associated with producing applications in a specific domain. Domain engineering identifies likely risks and associated aversion strategies which are supplied to application engineering.

In addition, domain engineering supplies project managers with adaptable planning documents that they tailor to direct their project through its increments for development and delivery of a product. This may be supported with process-driven tooling for monitoring progress against a tailored plan.

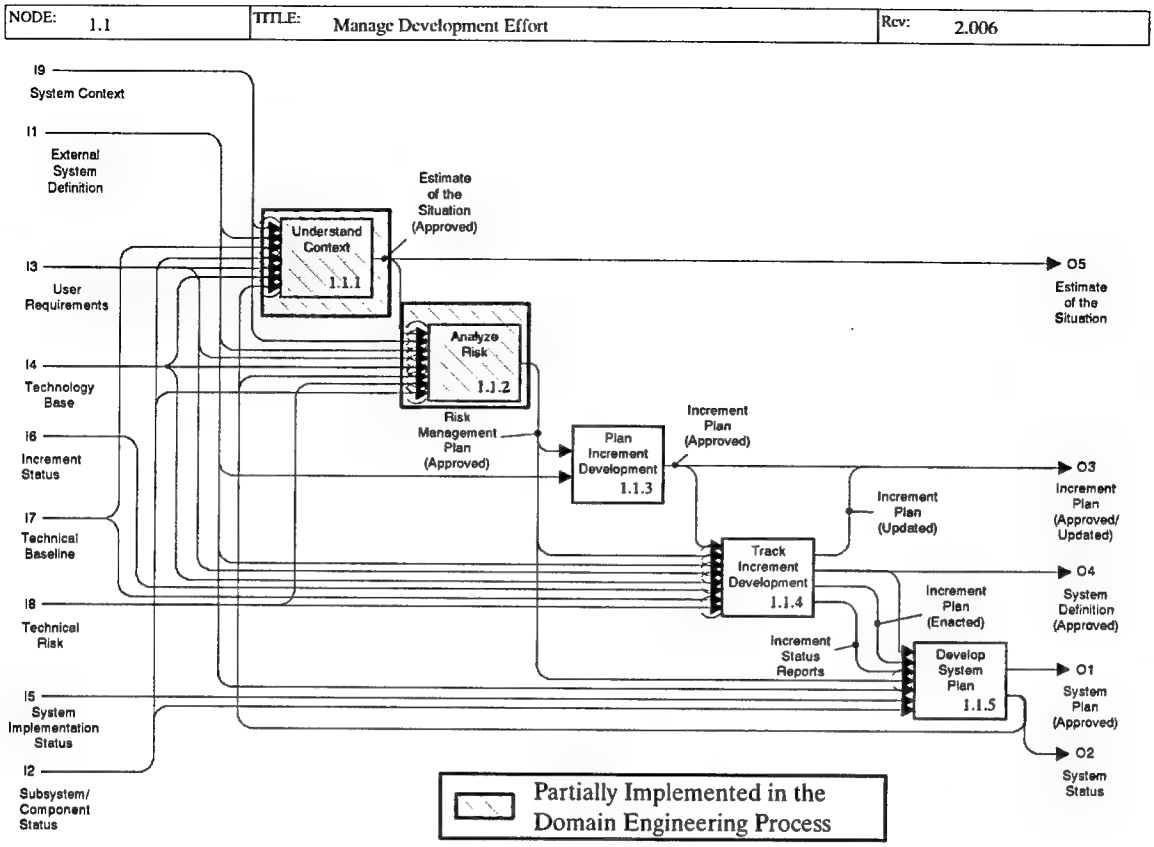


Figure 40. Manage Development Effort (Application Engineering Process)

5.3.2.2 Tailoring Technical Activities for Application Engineering

Domain engineering tailors the technical activities of the application engineering process by providing automated support for all or parts of each activity and providing accompanying adaptable assets that are used in generating required work products for each activity (see Figure 41). The following list describes how GSEP technical activities are tailored to define the application modeling portion of application engineering:

- ***Analyze Needs Activity (1.2.1).*** Domain engineering tailors this activity to fit the domain which constrains the possible stakeholders, problems, environments, and informal functions that are supportable. This activity is supported by automation that assists the application engineer to work with the customer to determine these factors within the framework of the domain.

- ***Define Requirements Activity (1.2.2).*** Domain engineering provides application engineering with a capability to describe (or model) a required system by making decisions that distinguish that system from other possible systems within the domain. These decisions are constrained to determine correlated, testable behavioral and performance requirements. Performance-related decisions, or ones that involve a trade-off between function and performance, may be partially deferred to the Evaluate Alternatives activity. Complete requirements for a system are a composite of requirements common to all systems in a domain, requirements decisions made in this activity, and requirements that are implied by decisions.

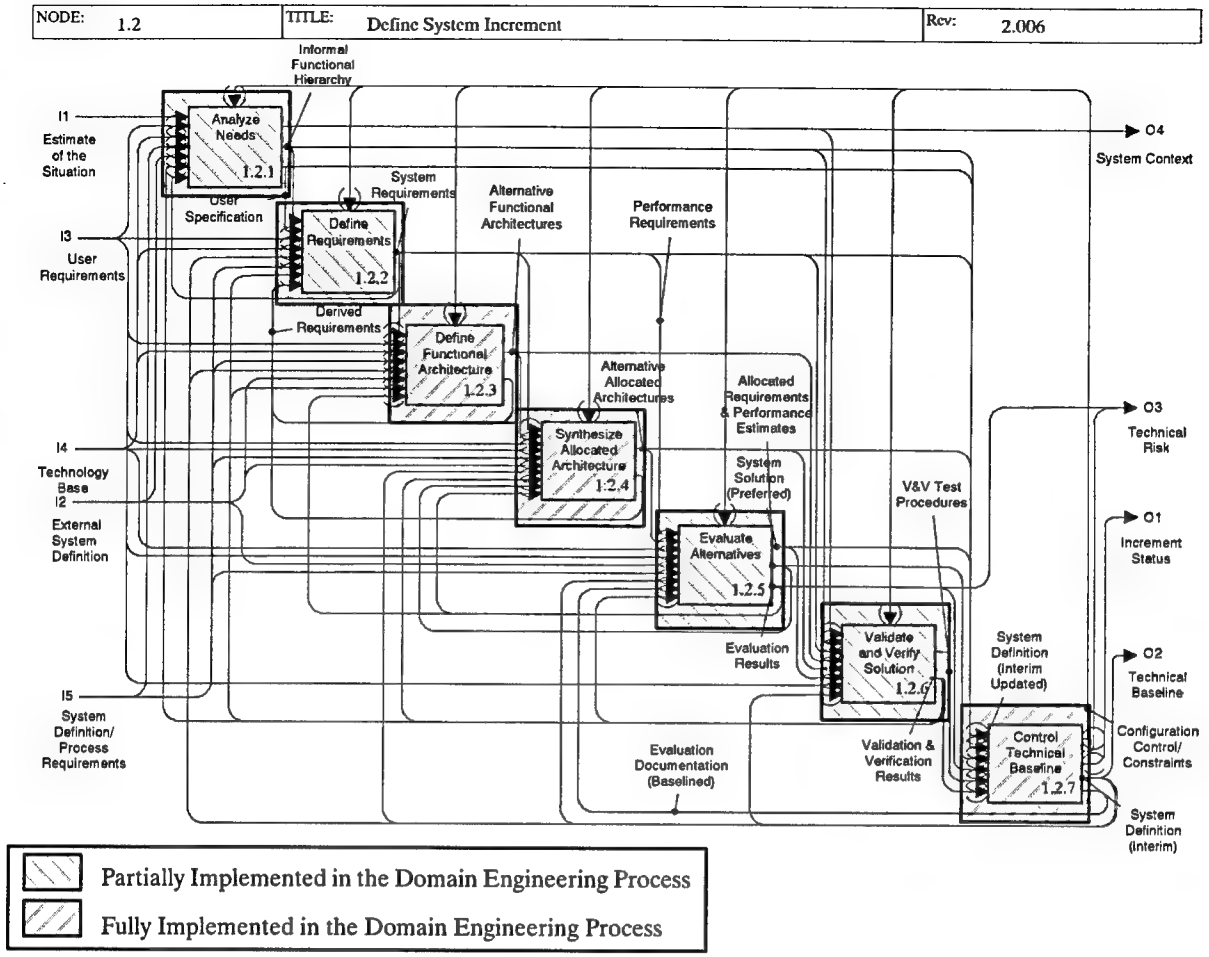


Figure 41. Define System (Application Engineering Process)

- ***Define Functional Architecture Activity (1.2.3).*** Domain engineering establishes a standardized functional architecture that is adaptable to supported requirements decisions (equivalent to alternative architectures). This activity is the mechanical generation of one or more functionally equivalent architectures. Each architecture partitions requirements into functions, identifies lower level functions, and defines the interfaces between functions to provide a solution structure appropriate to the requirements identified during the Define Requirements activity.

- ***Synthesize Allocated Architecture Activity (1.2.4).*** Domain engineering establishes a standardized allocated architecture that is adaptable to supported requirements decisions, consistent with the standardized functional architecture. This activity is the mechanical generation of alternative allocated architectures for each functional architecture. Each allocated architecture maps a functional architecture into a possible solution, determines its interfaces and technical parameters, integrates designs of lower level components, and determines a physical architecture based on the specific requirements identified during the Define Requirements activity.

- ***Evaluate Alternatives Activity (1.2.5).*** Domain engineering provides adaptable models to be used in this activity to evaluate the alternative allocated architectures and select a preferred system definition. Models provided address the performance analysis disciplines, sensitivities

to variations in system parameters and external environment interactions, technical risks and problems, and trade-offs characteristic of systems in the domain. Analyses are semi-automated with user facilities for controlling models and evaluating results.

- ***Validate and Verify Solution Activity (1.2.6).*** Domain engineering provides adaptable test procedures to be used in this activity to validate and verify the work products of other activities and of an implemented system solution. Domain engineering also provides adaptable procedures for executing and evaluating test results. Performance of validation and verification is semi-automated with user facilities for controlling solution analysis or execution in real or simulated target environments and for evaluating results.

- ***Control Technical Baseline Activity (1.2.7).*** Domain engineering builds automatic baselining of all decisions and application work products into the application engineering environment. New versions of affected work products may be produced whenever requirement and engineering decisions are made in the Analyze Needs, the Define Requirements, and the Evaluate Alternatives activities or as a side-effect of any activity that produces derived information (such as test results recorded during the Validate and Verify Solution activity).

5.4 AN EXAMPLE OF ENGINEERING FOR A PRODUCT FAMILY

To illustrate the concept of engineering for a product family, this section discusses an example of a product family corresponding to a hypothetical line of utility vans to be built by the passenger vehicles division of an automotive manufacturer. This discussion will focus, in overview, on the needs that motivate this family and how a product family orientation influences systems engineering as coordinated domain and application engineering efforts. Management activities of domain and application engineering are much the same as management in other GSEP-based processes. Therefore, this example is simplified to discuss only technical activities in more detail, recognizing that management is a necessary and integral part of the actual process.

In this example, domain engineering is responsible for creating the capability to produce and deliver any van that is part of the product family described here. That capability includes provision of marketing materials, a manufacturing facility, and supplies of component parts and raw materials. Application engineering comprises the interactions between a customer and a dealer to create an order for a particular van and between the dealer and a manufacturing representative who effect production and delivery of that van to the customer.

In terms of the Automobile System example discussed in Section 2, this section discusses primarily issues related to the operational (car) subsystem. However, domain engineering is normally concerned with all life-cycle subsystems, including operational, manufacturing, support, and others as appropriate, for a family of systems. Domain engineering may create a life-cycle subsystem in either a static (nonvarying) form for the entire product line or adaptable to the needs of each project. For example, the manufacturing subsystem of a domain, which is a major component of the application engineering environment (for producing individual vans to fill specific orders), may exist in only one form that is the same regardless of the characteristics of a particular van being produced or may be adaptable so that each project can create a more efficient facility tailored to their particular needs.

In the example, the passenger vehicles division has been producing such vans in three limited-option models for several years and wishes to offer greater variety to its customers in the future. To do this,

it is setting up an order-driven production facility that will allow flexible optioning of features by its dealers for customers. The targeted invoice price range for these vans is $25,000 to $40,000.

The utility vans product family will have a standard frame, but most other features will vary, though only in limited ways in many cases. Based on marketing surveys of customers and dealers, there is a market for vans with the following options (along with others not listed here):

- Any combination of fixed and convertible seating and cargo space, allowing for up to nine adult passengers with minimal storage, one passenger with maximum cargo, or anything in between
- A cargo storage infrastructure designed to the customer's specifications as to number, size, and positioning of compartments, within the constraints of cargo space determined above
- A choice of engines that provide different performance based on their power and efficiency characteristics
- An ability to emphasize fuel economy (up to at least 30 mpg), flexibility of usage (such as being able to tow a trailer), or safety (such as reinforced side panels), recognizing that conflicts among choices need to be identified for the purchaser

Some of these variability features may translate into adaptability of a single product, such as selection of the option to be able to convert between seating and cargo space; some options, such as fixed seating or engine selection, are strictly production options (that is, determined when a particular vehicle is ordered and then produced; other options, such as fuel economy, are implied by other features (vehicle weight and engine selection in the case of fuel economy).

5.4.1 Domain Engineering of the Utility Van Domain

Based on the above objectives, domain engineering in the Analyze Needs activity defines a domain of utility vans for the division. It performs an analysis of viability that determines that the division has the technical expertise to be successful and that there is a market adequate to support distribution of 100,000 units per year by this company. This is sufficient to generate profit based on the necessary investment in the domain and projected unit cost profile for a van.

In the Define Requirements activity, domain engineering systematically elaborates the requirements for the family of vans to be developed, taking account of how variability features correspond to different needs. Selected dealers and senior engineers of all disciplines participate to ensure completeness and accuracy.

In defining and evaluating adaptable forms of the functional and allocated architectures for vans, domain engineering cannot identify adequate current technology that will enable any van to be produced with a fuel economy of 30 mpg. Available engines and the minimum obtainable weight dictates a maximum of 28 mpg. Accordingly, this information is fed back to the Define Requirements activity for revision of requirements for the van family.

In defining the application engineering process and environment for the utility vans domain, domain engineering defines the driving procedures, maintenance, manufacturing, distribution, marketing, and recycling life-cycle subsystems appropriate to that domain. These subsystems may be made adaptable to variabilities in the product family leading to product variations such as different maintenance

schedules and procedures for vans with high-performance versus standard engines. The marketing subsystem provides dealers with a capability to help customers configure a van to suit their needs, verifying that no manufacturing constraints are violated. This includes an ability to generate realistic pictures of a van as specified by the customer and a simulation capability to evaluate trade-offs in expected operational usage.

To verify the domain, experienced engineers evaluate all work products during the domain engineering effort. To validate the domain, dealers and engineers, both ones who were involved in the requirements phase and others who were not, evaluate the van production capability as it evolves. This includes use of early versions of all subsystems, particularly the marketing subsystem. As the capability matures, prototype products are produced and evaluated, leading to refinements. Evaluations continue even after full production is active to ensure that the domain evolves to remain viable to all concerned.

5.4.2 Application Engineering in the Utility Van Domain

Application engineering, being standardized across the utility van product line and heavily automated, is a rapid and highly responsive process. The dealer and customers have almost direct control over the production and delivery of a highly tailored product. Dealers and customers are presented with a series of restricted questions that determine the options to characterize a particular, producible van. Decisions are made and changed as the customer's understanding of the alternatives for an end product improve.

The customer's decisions on options are assisted with an evaluation capability to view a simulated model of the vehicle to be produced. The model supports observation of both appearance and performance characteristics under a variety of anticipated operating conditions so that the customer can weigh the implications of alternative decisions. Although some feel overwhelmed by too many choices, dealers help them by fitting them into a few traditional categories of need and making many of the choices accordingly; most find the ability to view realistic models of the finished product helpful.

When the customer is happy with the particular van's characteristics, an order is generated for the dealer and transmitted to the factory. A van is produced to satisfy the specified options and delivered. At the same time, an owner's manual, including tailored driving procedures and maintenance instructions matching the van's specific features, is generated for the customer.

Although this example shares much in common with conventional manufacturing practices, it differs in its explicit and systematic focus on standardizing the product line upon common and variable features in products. Much more of the process is partially or fully automated. Consequently, support for much more extensive diversity in the product line becomes feasible, with much greater emphasis on allowing the customer to obtain a personally tailored product. Nevertheless, some things do not change; getting the best price still requires the customer to get bids from competing dealers.

5.5 TAILORING THE PRODUCT FAMILY PROCESS

The degree to which a product family process actually exhibits the features called out in Section 2.4 depends on the needs and abilities of the adopting organization. A GSEP-based Synthesis process of engineering for a product family should be tailored to suit the needs of each particular organization.

One form of tailoring is based on Consortium work in process improvement, specifically in reuse adoption. The *Reuse-Driven Software Processes Guidebook* (Software Productivity Consortium 1993a)

describes alternative forms of a Synthesis process which derive directly from critical success factors of reuse adoption. Any of these forms of tailored processes can be built around the GSEP. The guidebook defines opportunistic and leveraged forms in detail.

An inherent option for tailoring a Synthesis process occurs in the need to choose specific development methods for management, requirements analysis, design, implementation, testing, etc. Both the Consortium's Synthesis methodology and the GSEP were conceived specifically to allow an organization the choice of its preferred methods and to orient process definitions to being tailored accordingly.

An important consideration in creating a tailored process is achieving a suitable balance between the amount of engineering done at the domain level and the amount done at the applications level. Shifting engineering from the applications level to the domain level results in more standardized architectures and greater reliance on reusable assets, reducing the effort required to produce a system. However, this shift is also accompanied by an increase in the constraints that domain engineering imposes on the application engineering process and a decrease in the flexibility of application engineering to accommodate system needs that were not anticipated by domain engineering. The shift of resources from direct, revenue-producing systems (i.e., application systems) to the indirect, revenue-producing system (i.e., a domain) must be justified by greater profitability, due to lower unit cost, reduced errors, and shorter time to deployment, on each system produced.

Construction of modern, complex systems is a major undertaking that often requires the coordination of many large groups of engineers with expertise in diverse fields. Some of these groups may be organizations in separate companies, working jointly or as subcontractors, to deliver the required system. Engineers follow a discipline of systems engineering, in part, to partition the problem and apply appropriate expertise to solve each facet of the problem most effectively. This partitioning into subsystems and components is compatible with engineering for product families, particularly when engineers are able to follow a systematic approach that results in similar partitionings of similar systems. Each subsystem may correspond to a member of a family of similar subsystems (i.e., a hardware, software, or human system). The required subsystem can, in turn, be developed as a one-of-a-kind system or, based on a domain of such subsystems, as an instance of a product family.

This page intentionally left blank.

6. CONCLUSIONS

This report presents several new ideas and concepts in systems engineering. However, the GSEP is still work in progress. The purpose of this section is twofold:

- To explain why the new ideas and concepts are important and how they can be applied
- To describe the future plans for enhancement and application of the GSEP

6.1 GSEP CONCEPTS AND IDEAS

There are several new concepts and ideas that are introduced in this report. In some cases, it is not that the ideas or concepts are themselves new, it is that they are being used or implemented in a new and different way or are being combined with other concepts and ideas in new and potentially interesting ways. The following sections describe these concepts and ideas and the potential benefits that can be derived from their use.

6.1.1 A Generic Process

The GSEP is a generic development process (see Sections 2.2.2 and 3.1.2). As such, it can be applied to a wide variety of systems engineering development efforts, but to be applied it must be tailored. (Tailoring the GSEP is described in Section 4.) The GSEP approach of defining a generic process and explaining how to create a specific process from the generic one provides all the benefits of more traditional processes that are specific in nature, but additionally provides the ability to customize the GSEP for specific applications. It is impossible to accurately predict the future and impossible to predetermine every possible process need that an organization may require. By taking a generic/tailoring approach, the GSEP is capable of meeting process needs that even its creators did not envision.

6.1.2 Balanced Approach to Integration of Management and Technical Activities

The balanced approach to integration of management and technical activities is one of the fundamental GSEP concepts. It is understandable that managers and engineers view systems engineering from their own perspectives and concentrate on the activities that are part of their sphere of interest. It is also traditional to view management activities as taking place predominantly in the beginning and end of the life cycle (e.g., during the planning and delivery phases) and the technical activities as taking place predominantly in the middle of the life cycle (e.g., requirements definition and design). Although these situations routinely occur, they may result in poor communication and coordination between managers and engineers. At the extreme, lack of communication and coordination results in the following:

- Plans that are not realistic and are not committed to by the engineers
- A lack of management during the development phases of the life cycle

- A lack of understanding and insight on the part of management into the technical roles
- A lack of understanding and insight on the part of the engineers into the management roles

The GSEP provides a process solution that ensures the participation of engineers in the management tasks throughout the system life cycle. Additionally, the GSEP has mechanisms that ensure management is aware of the technical issues and concerns and that management and engineers work together to develop the system. In GSEP, the management activities are performed continually throughout the system life cycle and emphasized just as much in the middle of the life cycle as in the beginning and end. Through the balanced integration of management and technical activities, GSEP provides the foundation for cooperation, communication, and mutual respect between managers and engineers.

6.1.3 Commitment/Stakeholder Involvement

Integration of management and technical activities alone is insufficient to ensure stakeholder involvement and commitment in the system being developed. Other mechanisms must exist to promote information exchange and achieving consensus among all affected parties. The GSEP provides a solution that is part process (i.e., defined activities) and part application of that process (i.e., how GSEP concepts can be implemented). The review and commitment activities are part of the GSEP process (e.g., Review Context [1.1.1.3] and Commit to Plan [1.1.3.4]). The purpose of these activities is to encourage information exchange and consensus building.

The IPT types described in Section 2.3.3 and 3.3.3 are not specifically part of the GSEP activity descriptions; however, the GSEP supports the formation and existence of IPTs. The GSEP has two activities, Define Approach (1.1.1.1) and Determine Stakeholders (1.2.1.1), that identify the total set of stakeholders. However, not every stakeholder takes part in every IPT. This is because there are potentially numerous stakeholders and not all of them need be in every IPT. IPTs are focused on a specific objective. Consequently, it is important to define the IPT objective(s) and identify the set of stakeholders (a subset of the entire set of stakeholders) that is appropriate to meet the objective(s).

Defining and focusing on system objectives are important parts of the GSEP process. These objectives are used to determine stakeholder participation and involvement. Processes that do not include the definition of objectives hamper efforts to establish focused, communication channels such as IPTs.

6.1.4 Mechanisms for Communication of Information and Coordination of the Effort

Successful development requires the timely communication of information and effective coordination of the development effort. The GSEP supports all three types of information flow described in Section 3.3.2.1. GSEP also defines coordination activities (e.g., Develop System Plan [1.1.5]) and establishes where and how coordination should be done. In the GSEP, coordination activities are done at a level where there is sufficient visibility. This means that higher level activities (i.e., system-level activities) have a broad, but shallow, view of the development effort and, consequently, they coordinate high-level activities (i.e., activities at their level or immediately below) and establish communications links that work at their level and the next lower level. Lower level activities have a deep, but narrow, view of the development effort, and they coordinate activities within their narrow scope and establish communications activities that pass information down to the lowest level in which they have visibility.

In tailoring (see Section 4.2.1.2), the GSEP is used to define the process architecture that specifies the interactions between the system, subsystems, and components. In defining this architecture, the GSEP calls out an approach to engineering the process that ensures a well-engineered and integrated systems engineering process.

6.1.5 Separation of Concerns

GSEP's emphasis on defining objectives provides the mean for achieving separation of concerns. (See Section 2.3.4 and 3.3.4 for a discussion on the separation of concerns.) In the GSEP, objectives for performing the development activities are clearly defined. This provides an understanding of what must be done and why, without overly constraining the creativity of the activity performers. This promotes participant motivation and buy-in. The more experience managers and engineers have with a process, like GSEP, that provides guidance but does not stifle their ingenuity, the more they appreciate the process and recognize its benefits.

One important separation of concerns issue that GSEP addresses is the intentional separation of the informal functional hierarchy and the functional architecture. Although the informal functional architecture defines a loose system design that is based on technical historical biases and customer/user preconceptions, this architecture should not constrain the system design. The objective of informal functional hierarchy is to derive the system requirements. Once the system requirements are defined, the alternative functional architectures should be created based on the requirements rather than the informal functional hierarchy. The functional architecture should be based on the relationship between the requirements rather than the historical biases and preconceptions. This is not to say that reuse of previous system components or architectures should not be considered, but rather that the functional architecture should not be constrained by reuse until the requirements are well understood and documented and alternative designs are under consideration.

6.1.6 Process Improvement Architecture

There are many approaches to implementing a process improvement process. They lie on a continuum with two extremes: including the process improvement activities in the development activities and having completely separate processes, one for development and one for process improvement. GSEP takes an approach in the middle of these two extremes by keeping the process improvement activities separate from the development activities, but not making the two processes completely separate (see Section 4.2.1.2).

The GSEP approach has an advantage over each of the two extremes. When the process improvement activities are not separate from the development activities (e.g., the activity that reviews the product being produced also reviews the process used to create it), it is difficult focus on the process improvement aspects of the activities. Typically, the process improvement work takes a "back seat" to the development work. Keeping the activities separate makes it easier to ensure that the process improvement work receives the attention it requires and is not overlooked. If the process improvement process is completely separate, it runs the risk of being sufficiently "out-of-touch" with the development process it is designed to improve. Keeping the process improvement and development processes closely linked ensures that process improvement has the information and feedback necessary to be successful.

6.1.7 Risk Driven

GSEP uses risk as a process driver. This means that the risk level is an important factor in decision making. Decisions such as what to do, when to do it, and who should do it, are all influenced by the

results of risk analysis. For example, if development of a part of the system is high risk, risk aversion/mitigation activities are preformed to bring the risk down to an acceptable level before a major commitment of time and resources is made to develop the part. This minimizes the amount of rework and associated frustration.

6.1.8 System Plan Creation

The System Plan guides the entire development effort. Consequently, the process that is used for its creation should have many of the same features as the system development process. In the GSEP, the GSEP itself is tailored to create the process for creation of the System Plan (see Section 4.2.1.1). This ensures that the System Plan is created with the rigor, thoughtful concern, and insight that should be given to a plan that will, potentially, determine the success or failure of the development effort.

6.1.9 Product Families

Although GSEP was not created specifically to support the creation of product families, it can be used for this purpose. (See Section 5 for a description of how GSEP can be used to create product families.) GSEP's ability to support domain engineering, as well as application engineering, proves its flexibility and makes it an appropriate choice for organizations who are considering the creation of product families.

6.2 FUTURE DIRECTION AND PLANS

This technical report contains work-in-progress. Like all processes, GSEP requires several process improvement cycles to help refine its concepts and provide additional detail in the activity definitions. Although this technical report describes GSEP in sufficient detail so that it can be implemented, work still remains to be done that will make GSEP more prescriptive.

The following tasks describe the next set of improvements to the GSEP:

- Enhance GSEP activity descriptions to include roles, required and optional resources, potential tool support, measurables, and validation and verification criteria
- Complete the mapping of the generated requirements for a systems engineering process to the GSEP to ensure complete requirements traceability
- Improve the integration of tool support for the GSEP activities
- Apply the GSEP on several, varied projects to provide the process improvement feedback necessary for making improvements

When more is understood about the GSEP, and as systems engineering trends are realized, the GSEP development team will update this report to reflect the increased knowledge.

APPENDIX A. GSEP IDEF DIAGRAMS

This appendix documents the IDEF0 diagrams that define the GSEP. See Section 3.1.4 to understand how to interpret the IDEF diagrams. Figure 42 outlines the diagramming conventions in its legends.

The GSEP defines a hierarchy of activities that are formed by a tree of IDEF0 diagrams. Table 1 shows the depth-first structure for the diagrams.

Table 1. IDEF Diagram Structure for Appendix A

Number of Diagram	Title of Diagram	Figure/Page Number
−0	Enterprise Context	Figure 42, Page 98
0	Develop Systems	Figure 43, Page 99
1	Develop System	Figure 44, Page 100
1.1	Manage Development Effort	Figure 45, Page 101
1.1.1	Understand Context	Figure 46, Page 102
1.1.2	Analyze Risk	Figure 47, Page 103
1.1.3	Plan Increment Development	Figure 48, Page 104
1.1.4	Track Increment Development	Figure 49, Page 105
1.1.5	Develop System Plan	Figure 50, Page 106
1.2	Define System Increment	Figure 51, Page 107
1.2.1	Analyze Needs	Figure 52, Page 108
1.2.2	Define Requirements	Figure 53, Page 109
1.2.3	Define Functional Architecture	Figure 54, Page 110
1.2.4	Synthesize Allocated Architecture	Figure 55, Page 111
1.2.5	Evaluate Alternatives	Figure 56, Page 112
1.2.6	Validate and Verify Solution	Figure 57, Page 113
1.2.7	Control Technical Baseline	Figure 58, Page 114

Appendix B describes the activity and gives entry and exit criteria for the activity. The work products that are input and output to and from the activity are also documented in Appendix B. Appendix C documents the work product definitions.

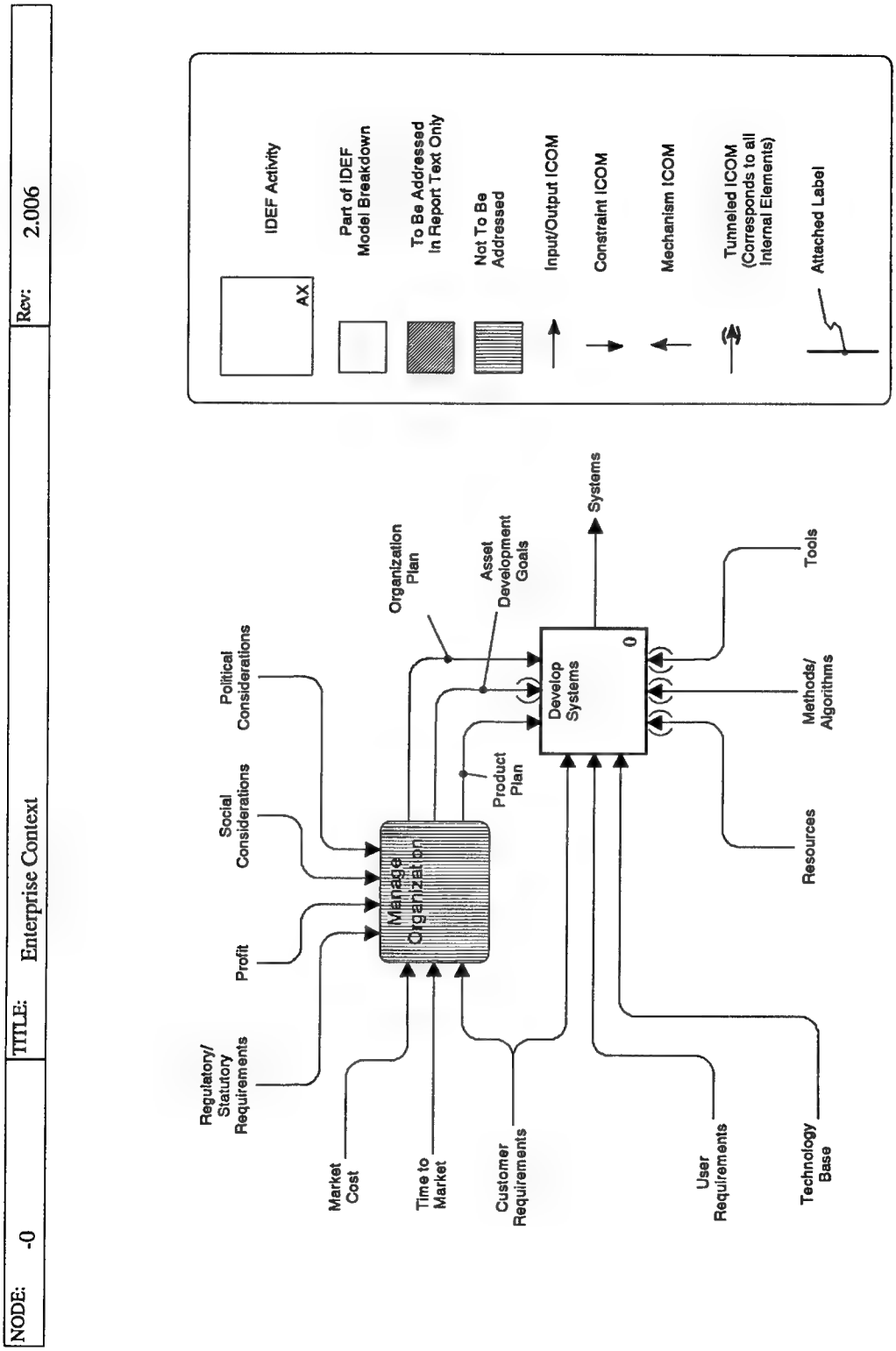


Figure 42. Enterprise Context

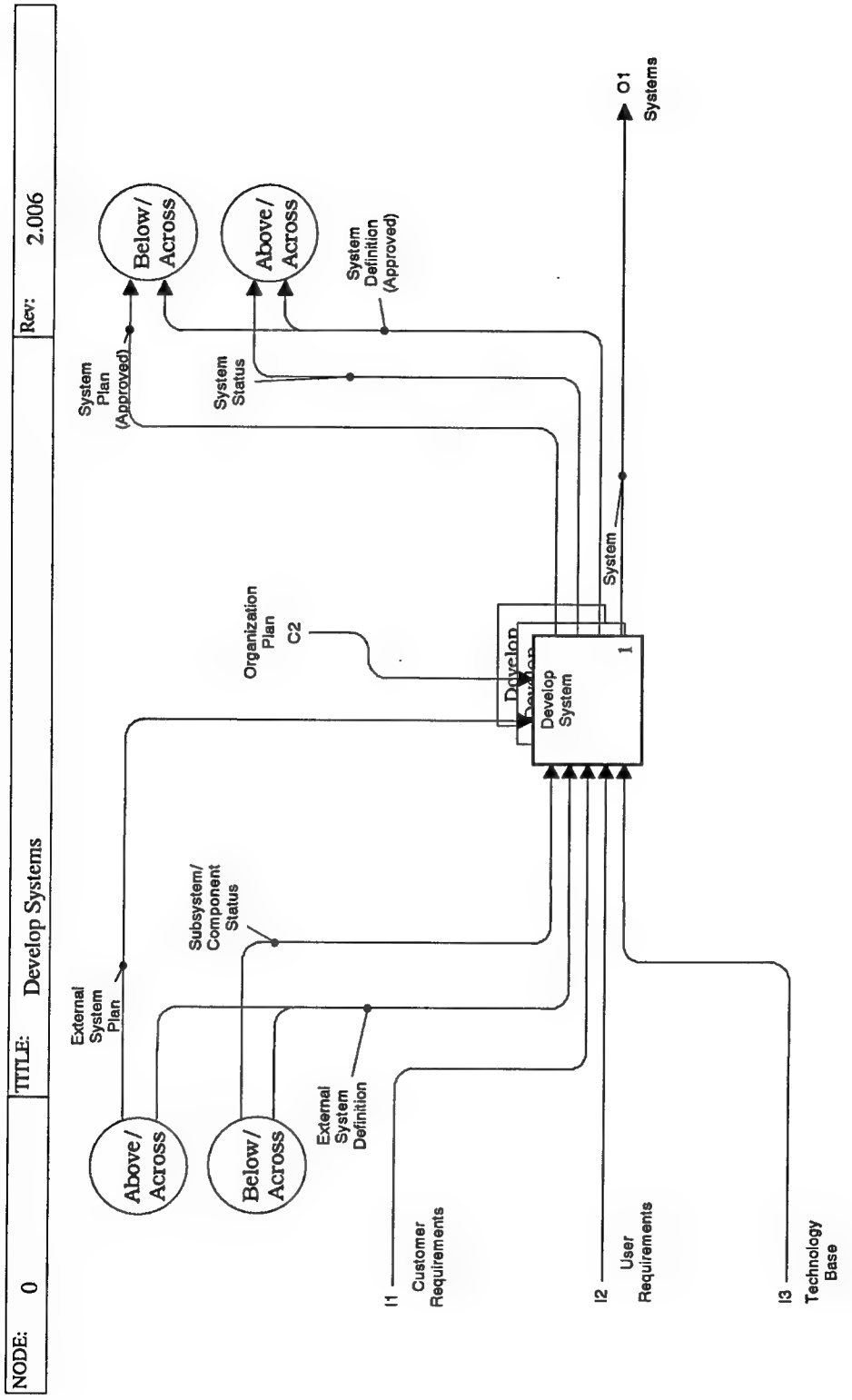


Figure 43. Develop Systems

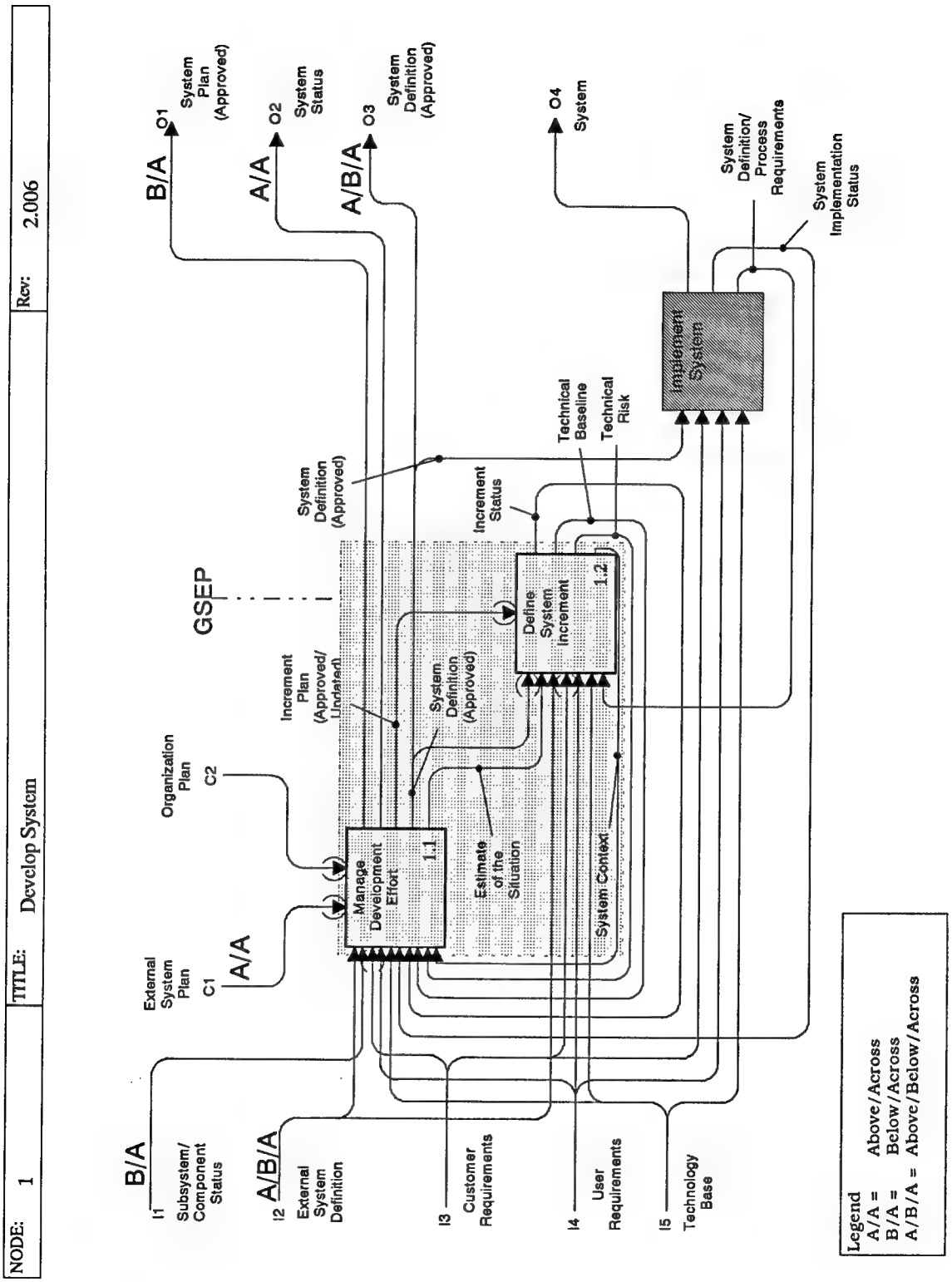


Figure 44. Develop System

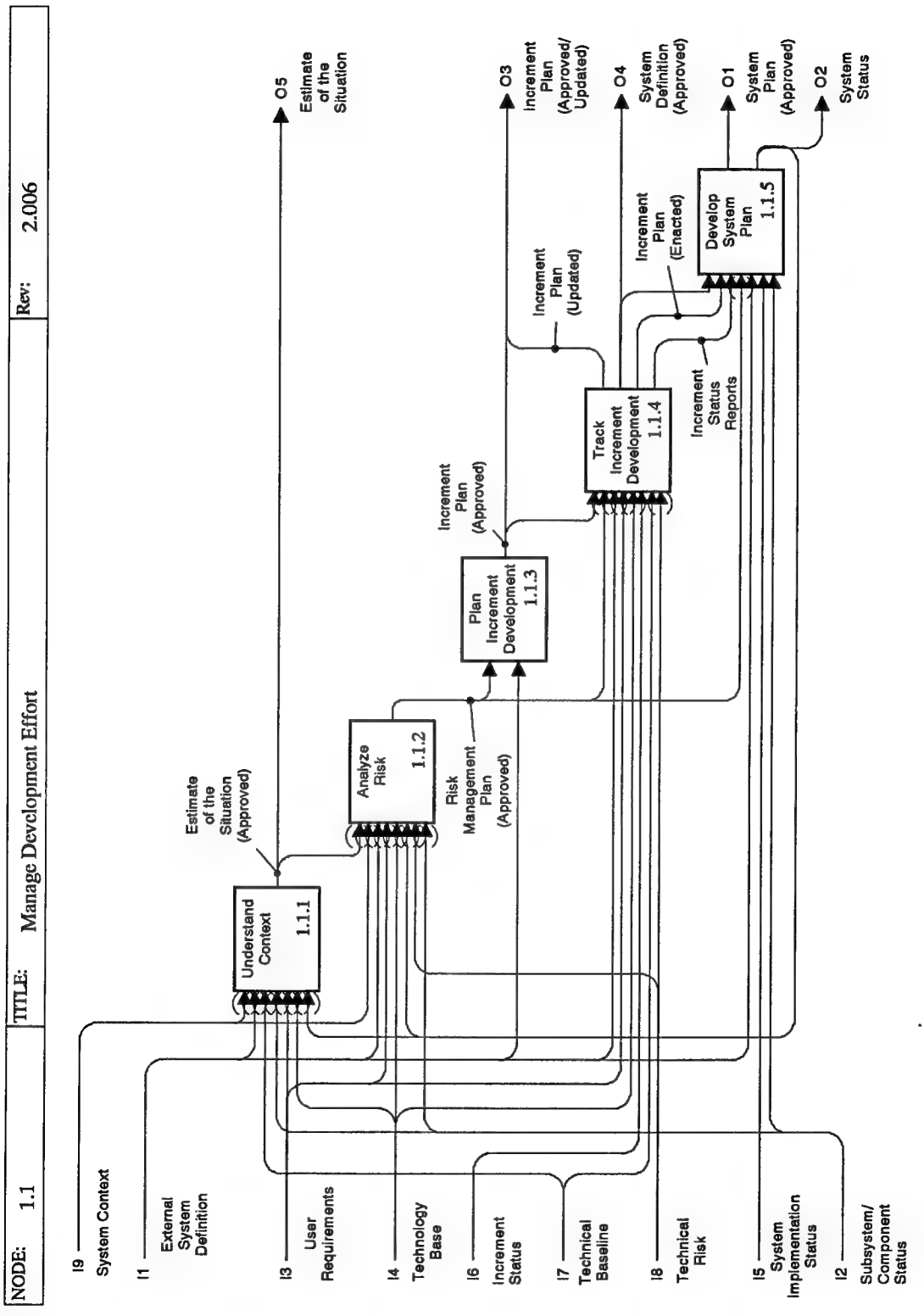


Figure 45. Manage Development Effort

NODE: 1.1.1	TITLE: Understand Context	Rev: 2.006

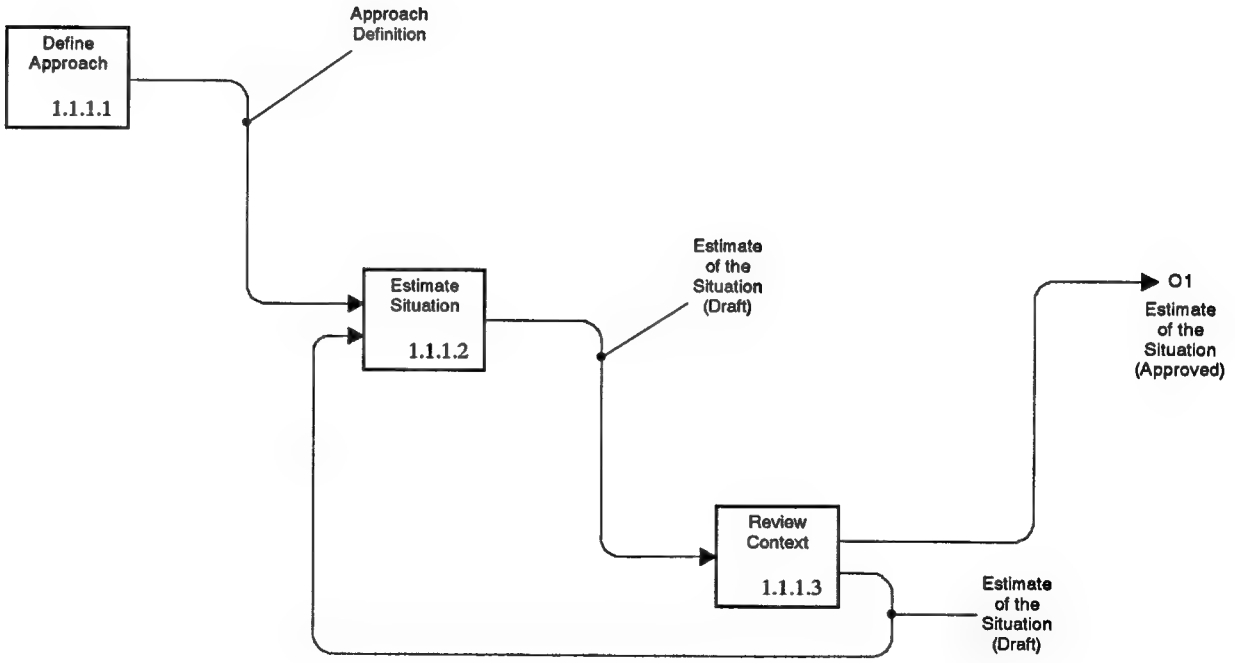


Figure 46. Understand Context

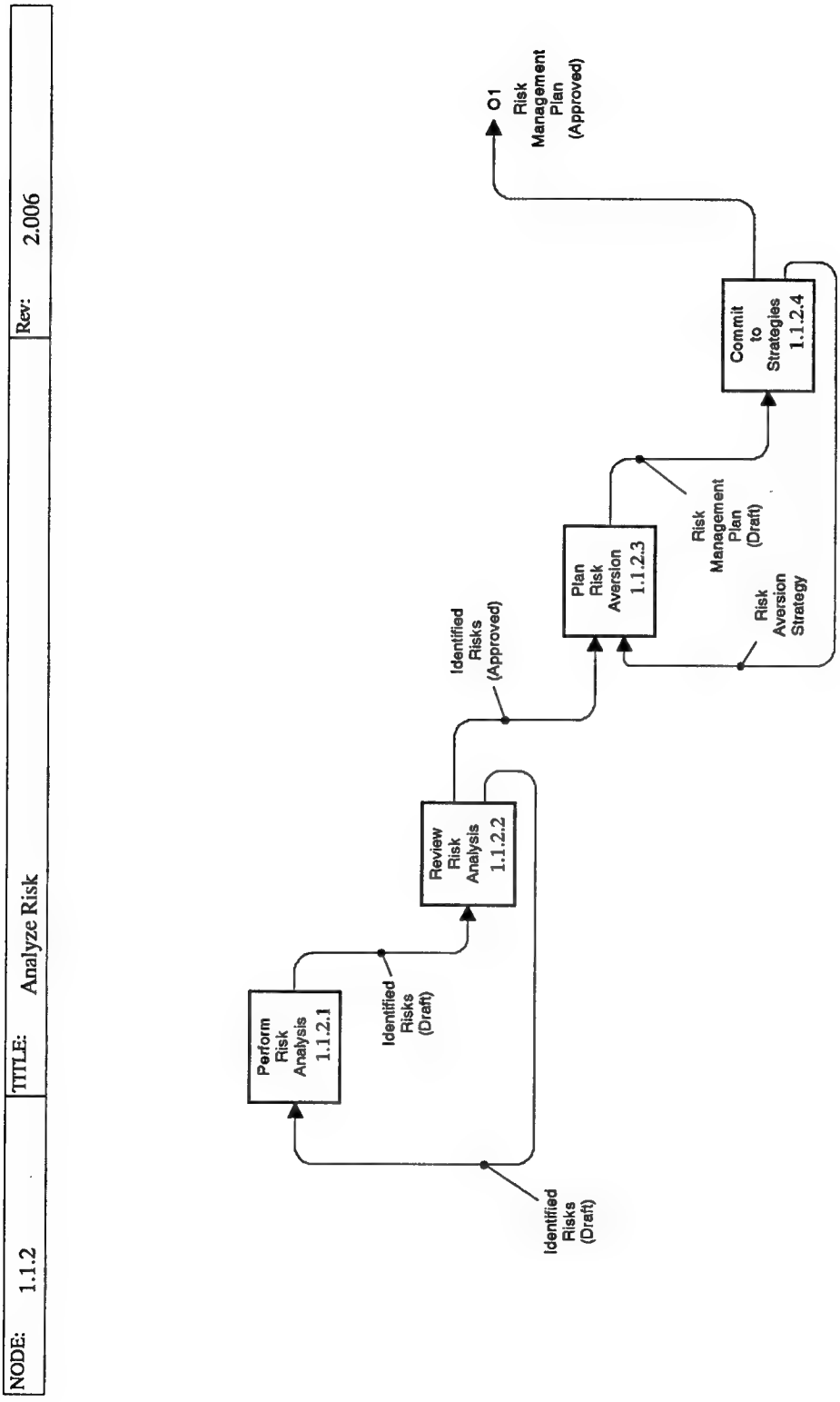


Figure 47. Analyze Risk

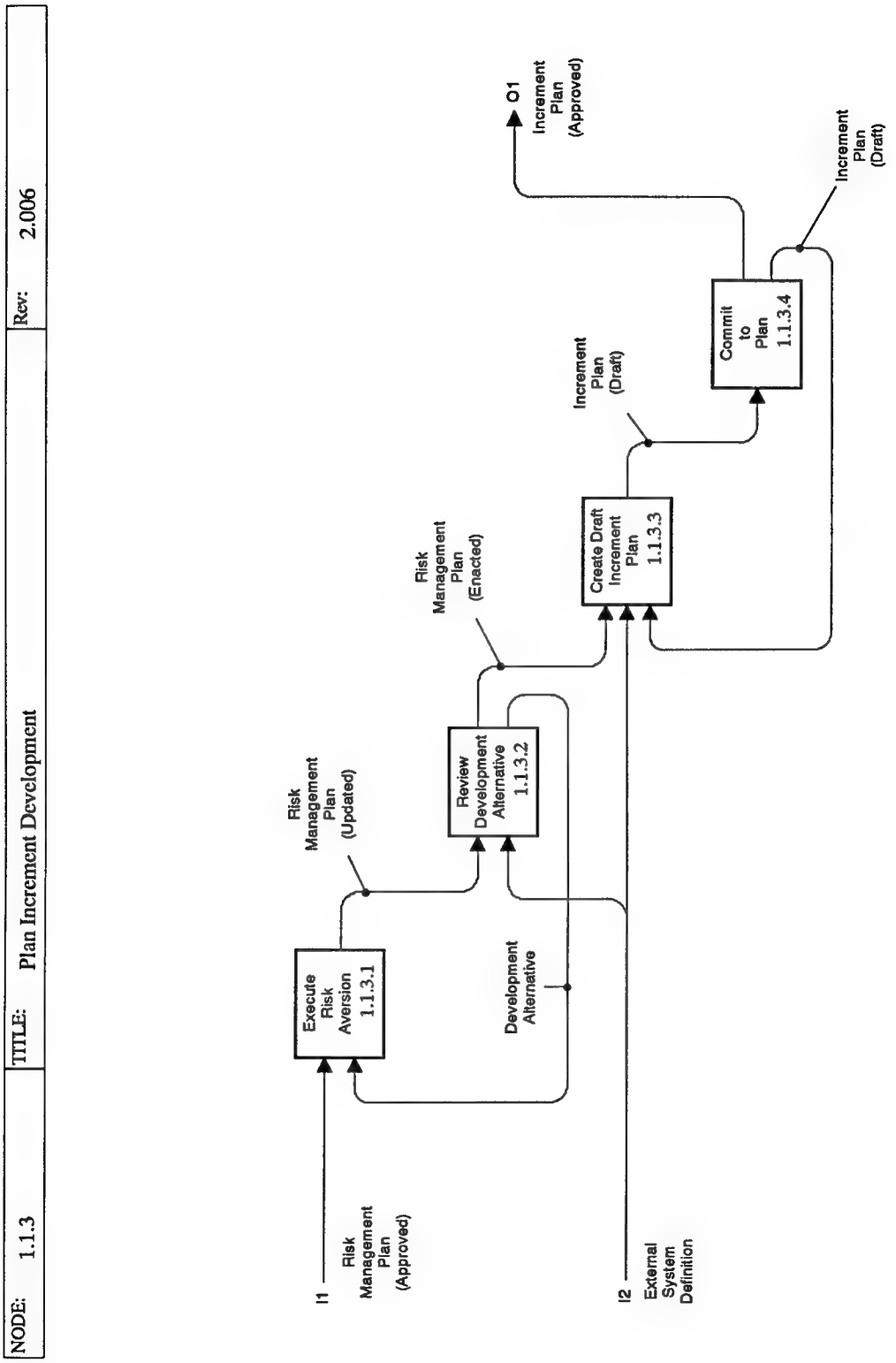


Figure 48. Plan Increment Development

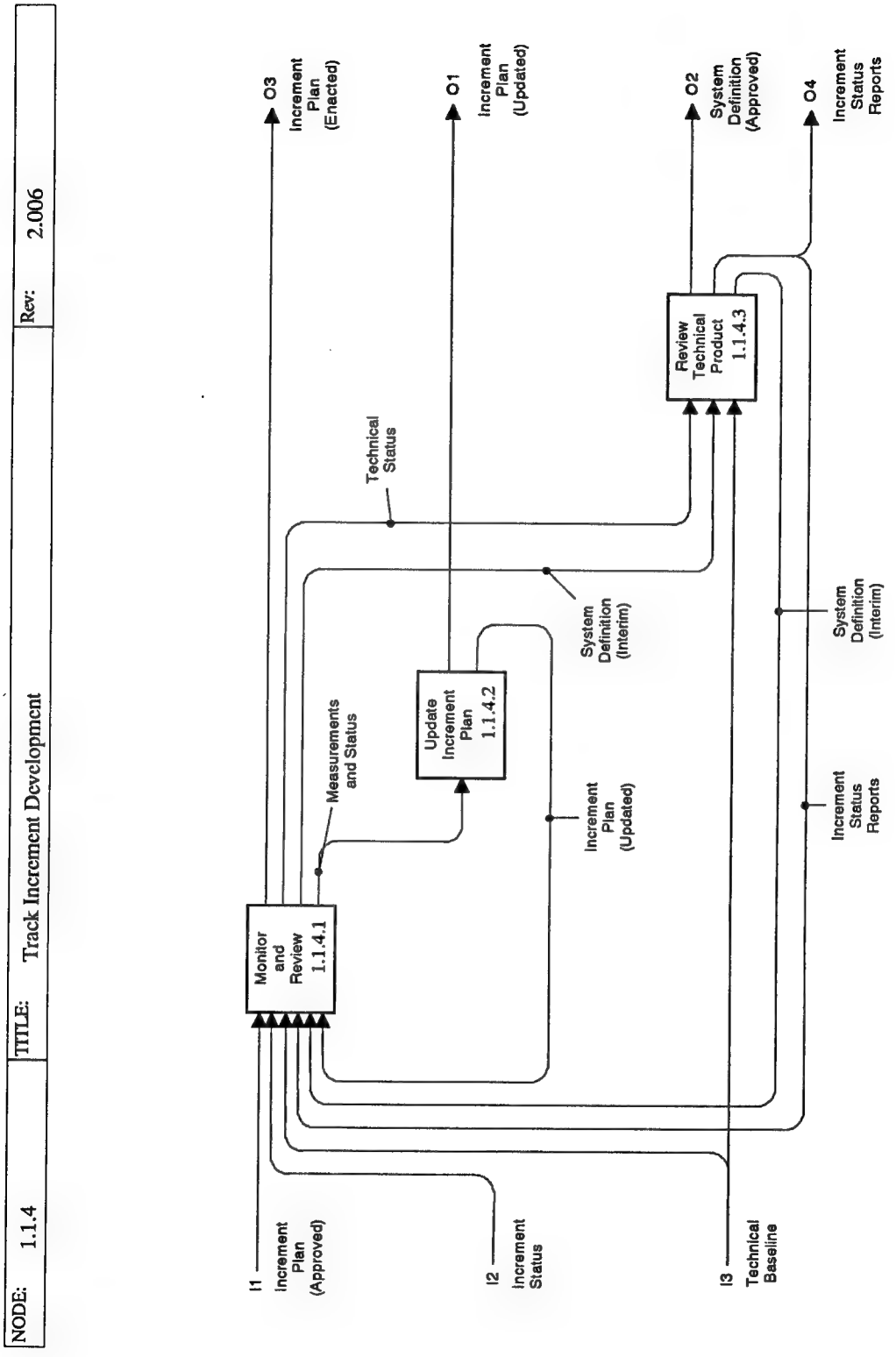


Figure 49. Track Increment Development

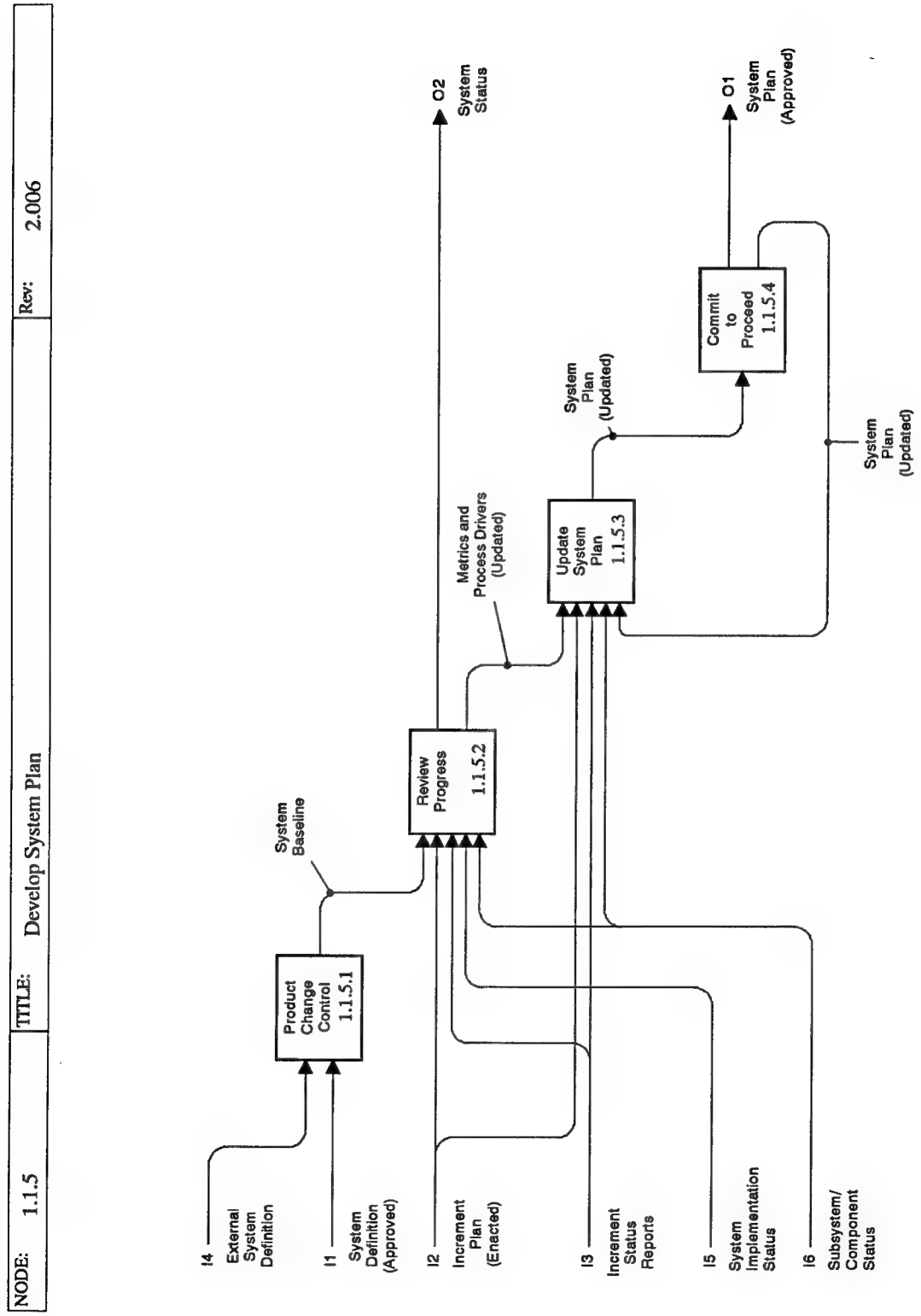


Figure 50. Develop System Plan

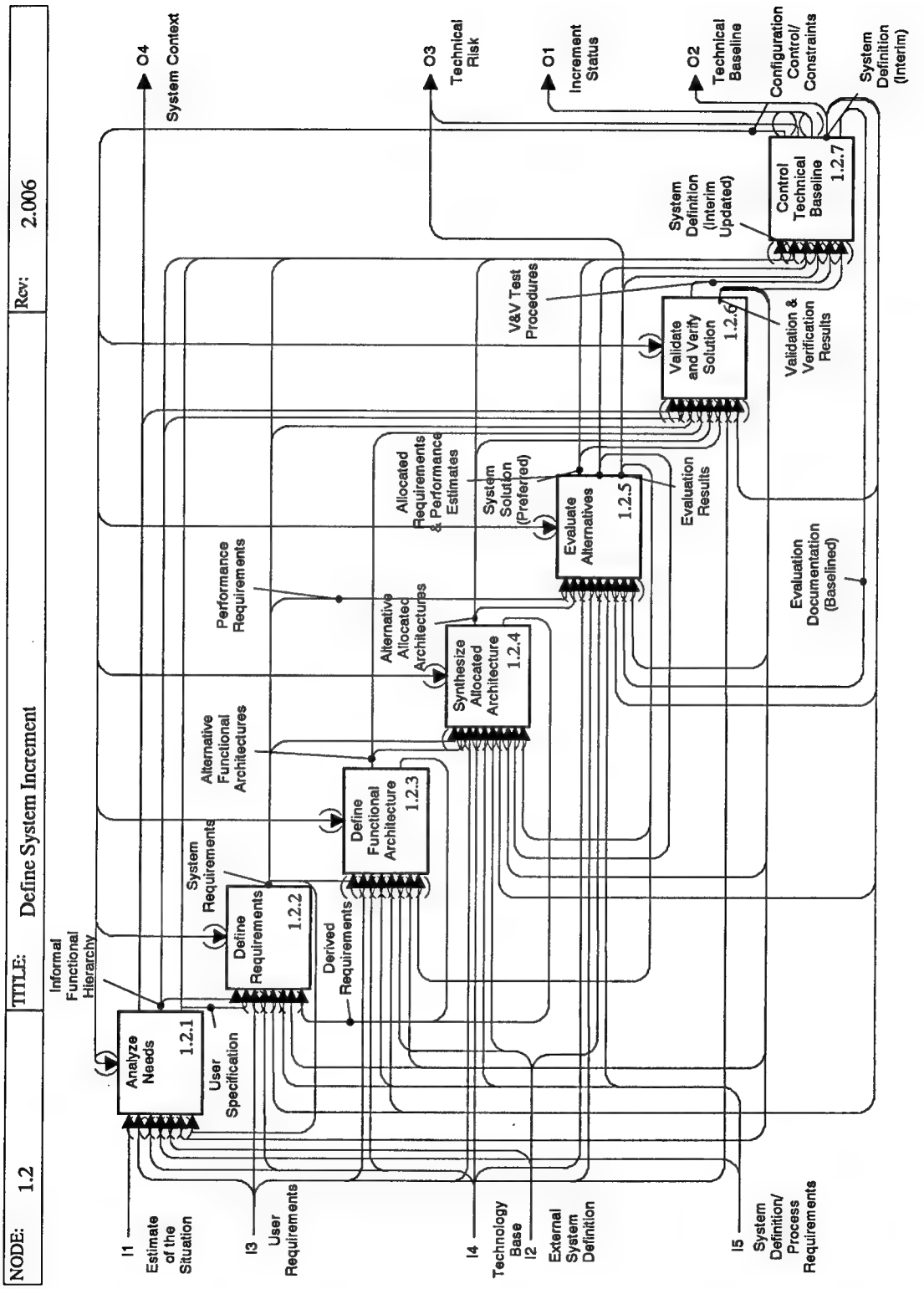


Figure 51. Define System Increment

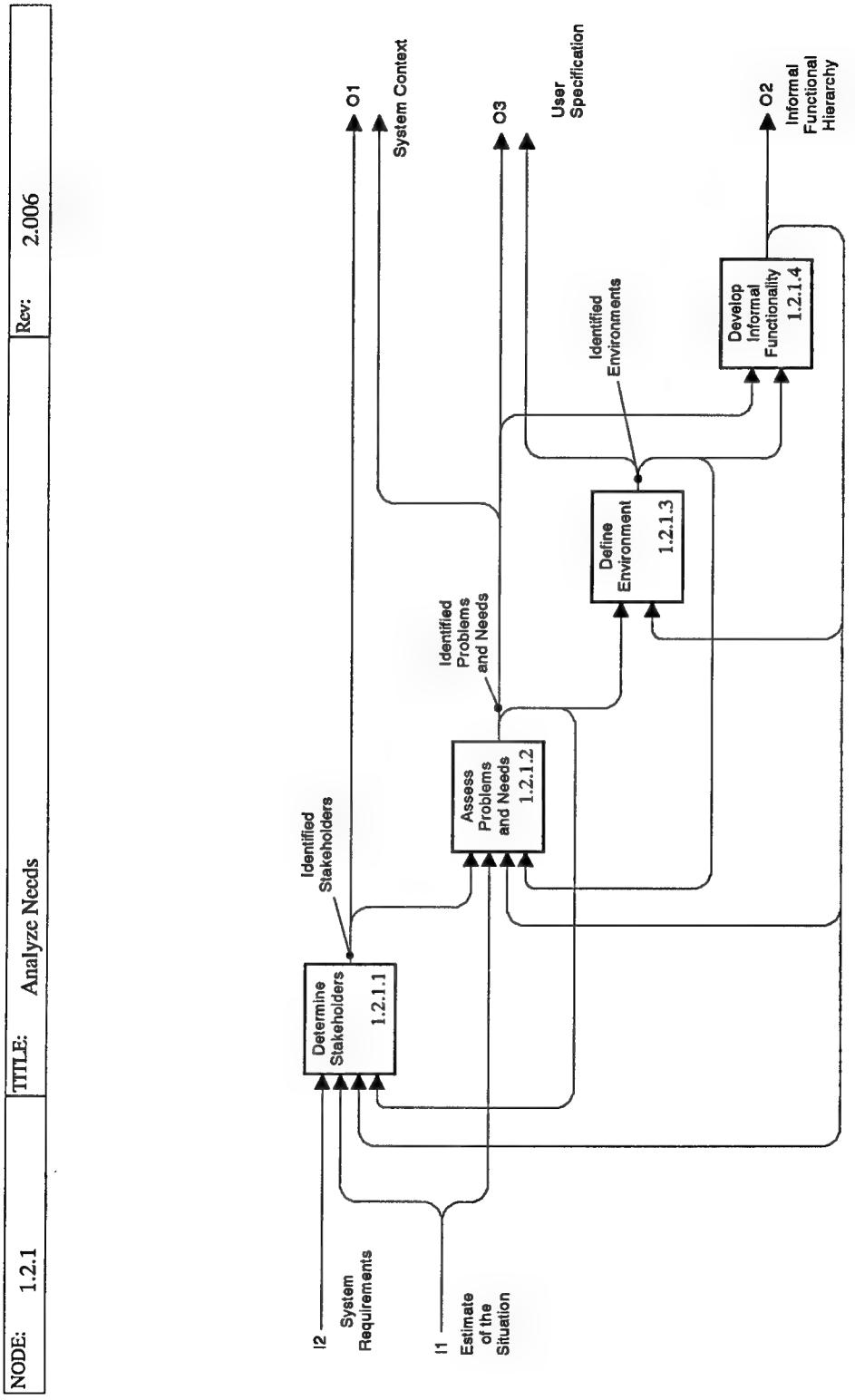


Figure 52. Analyze Needs

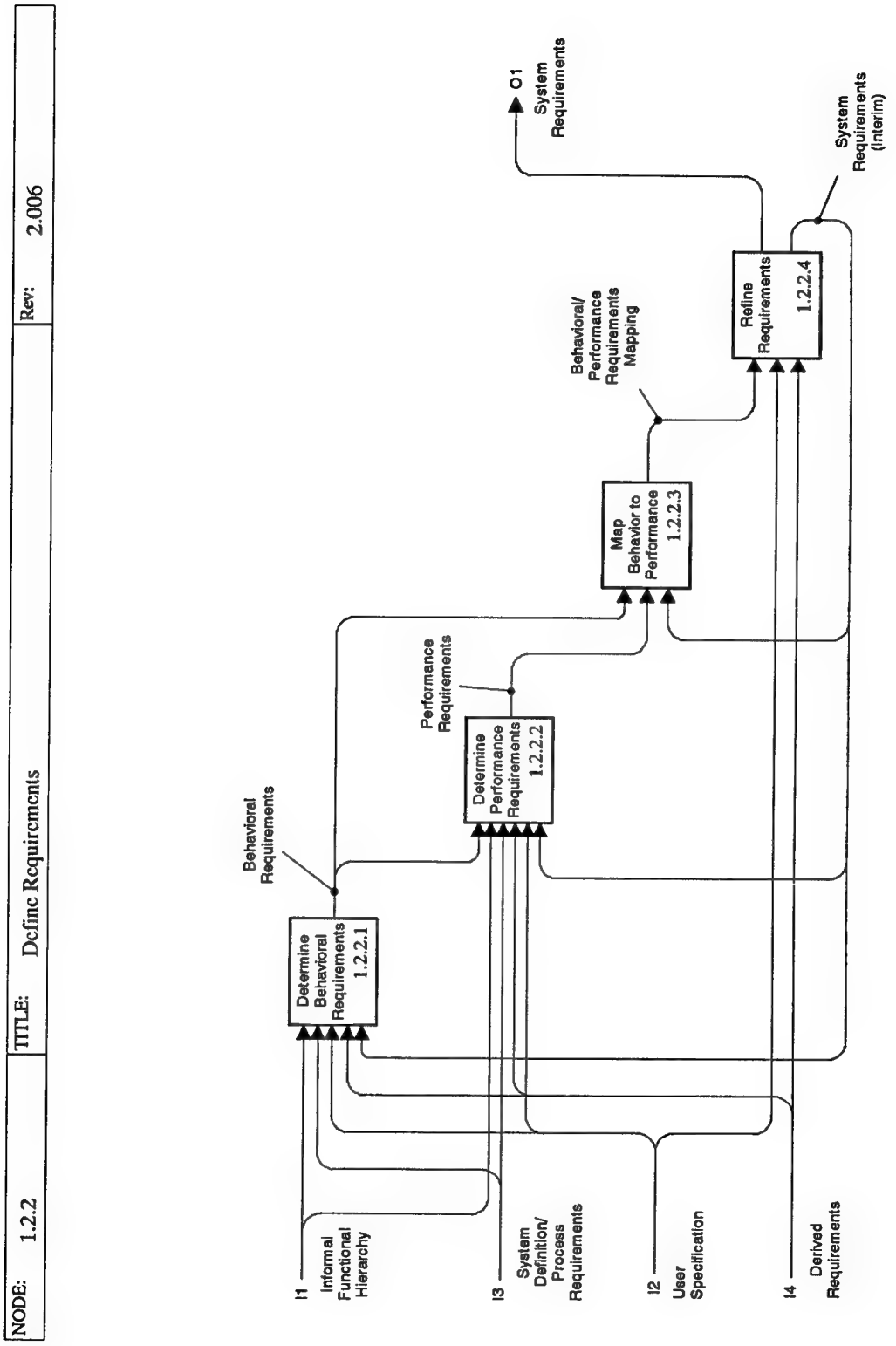


Figure 53. Define Requirements

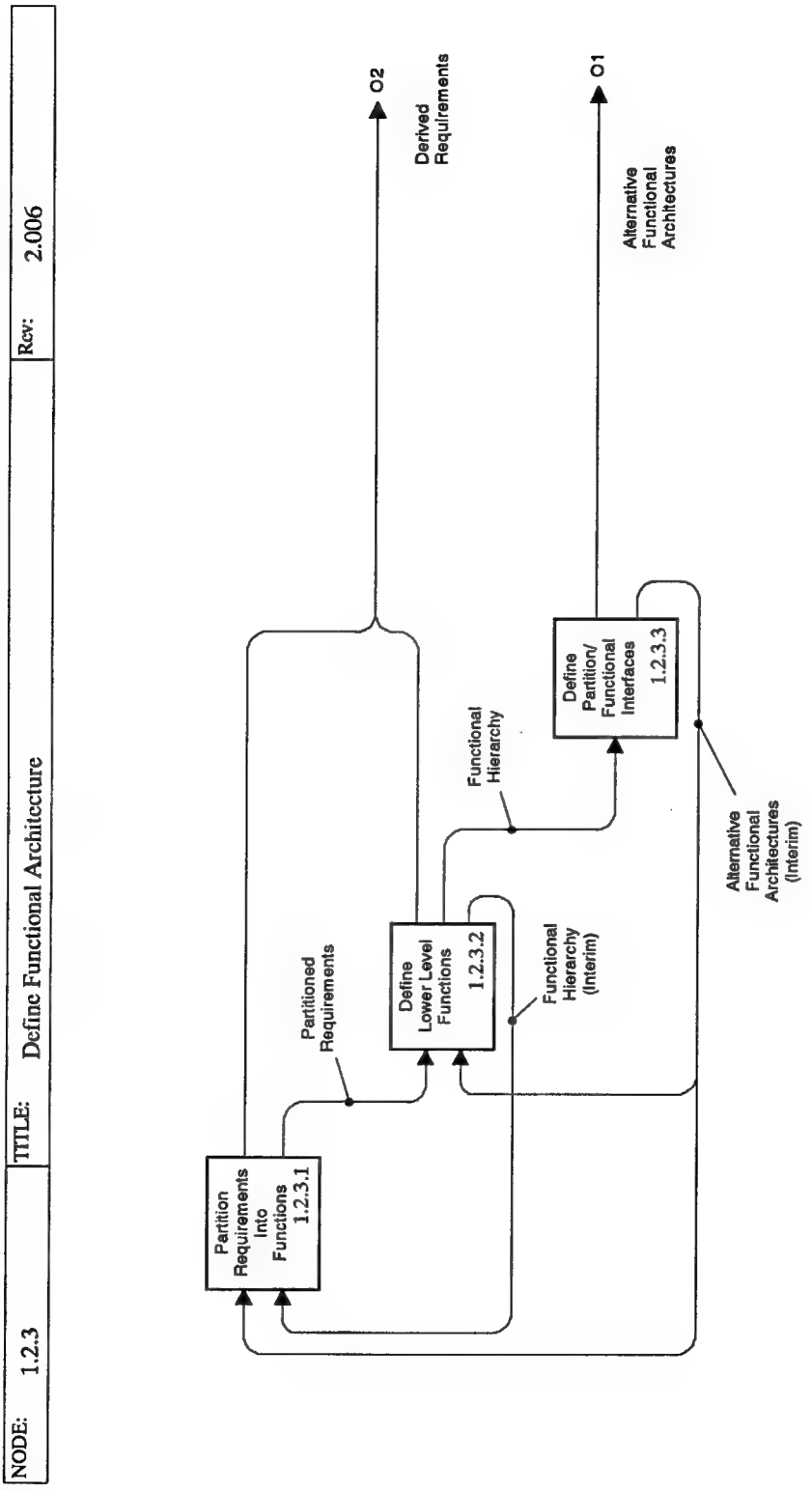


Figure 54. Define Functional Architecture

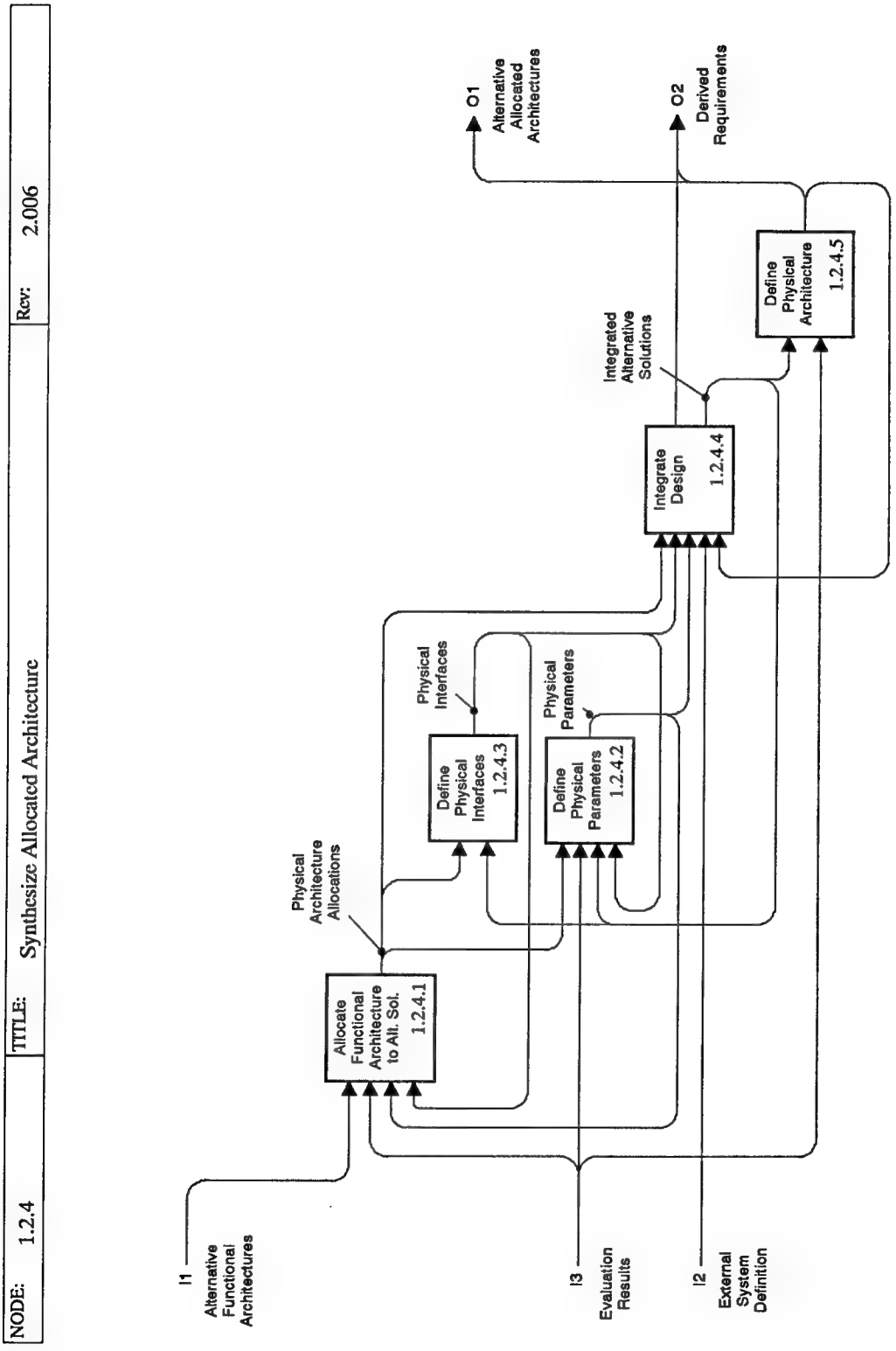


Figure 55. Synthesize Allocated Architecture

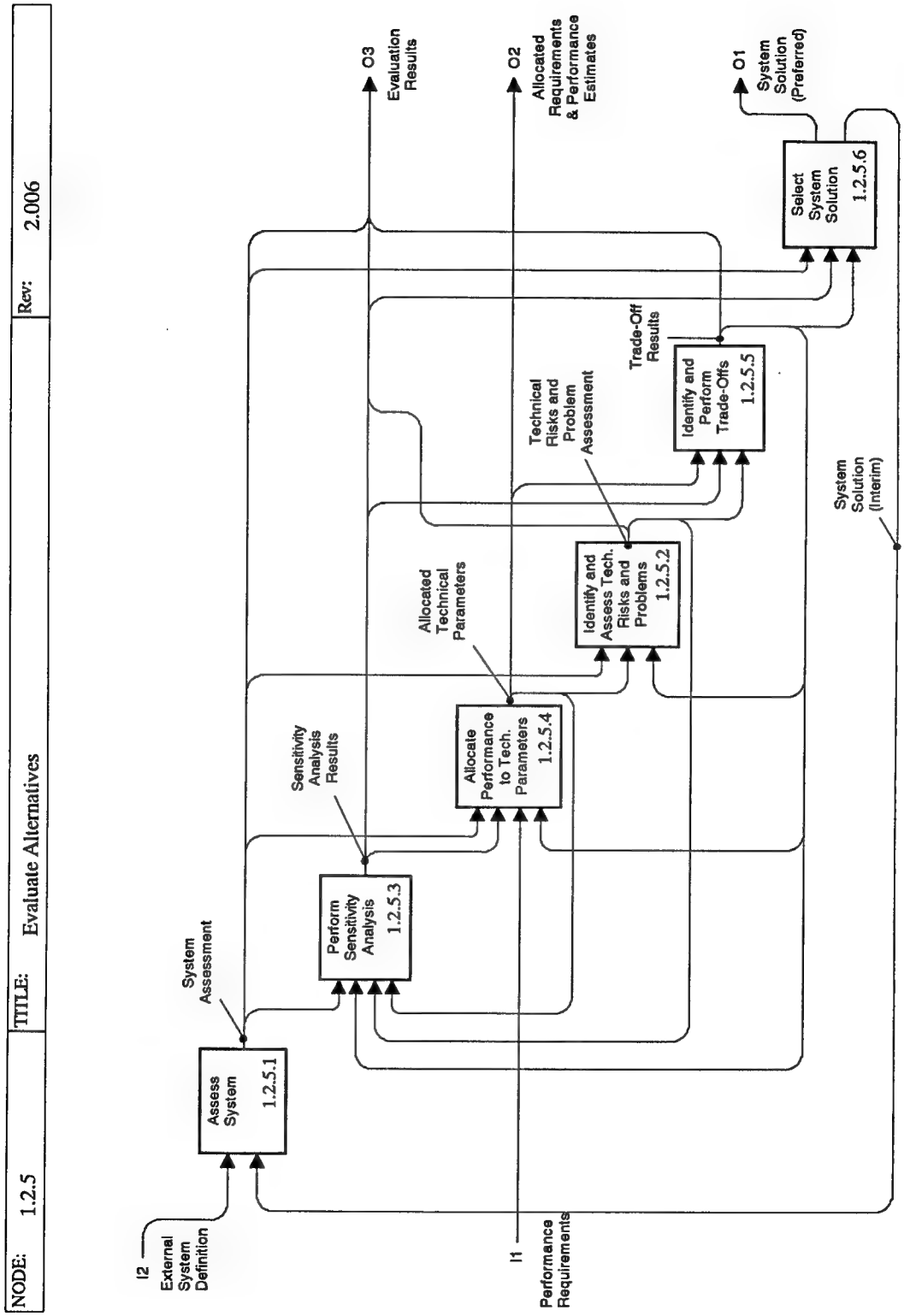


Figure 56. Evaluate Alternatives

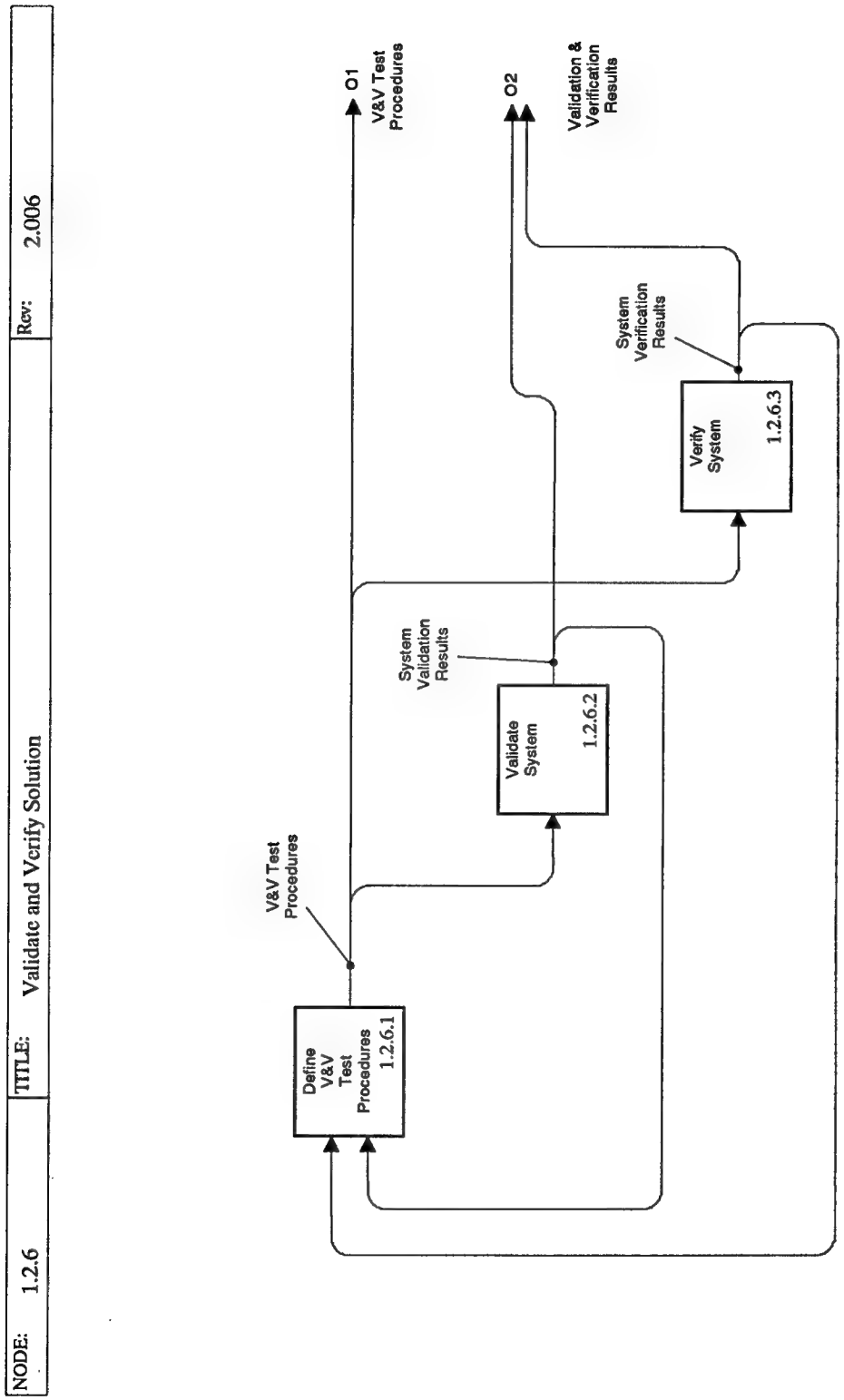


Figure 57. Validate and Verify Solution

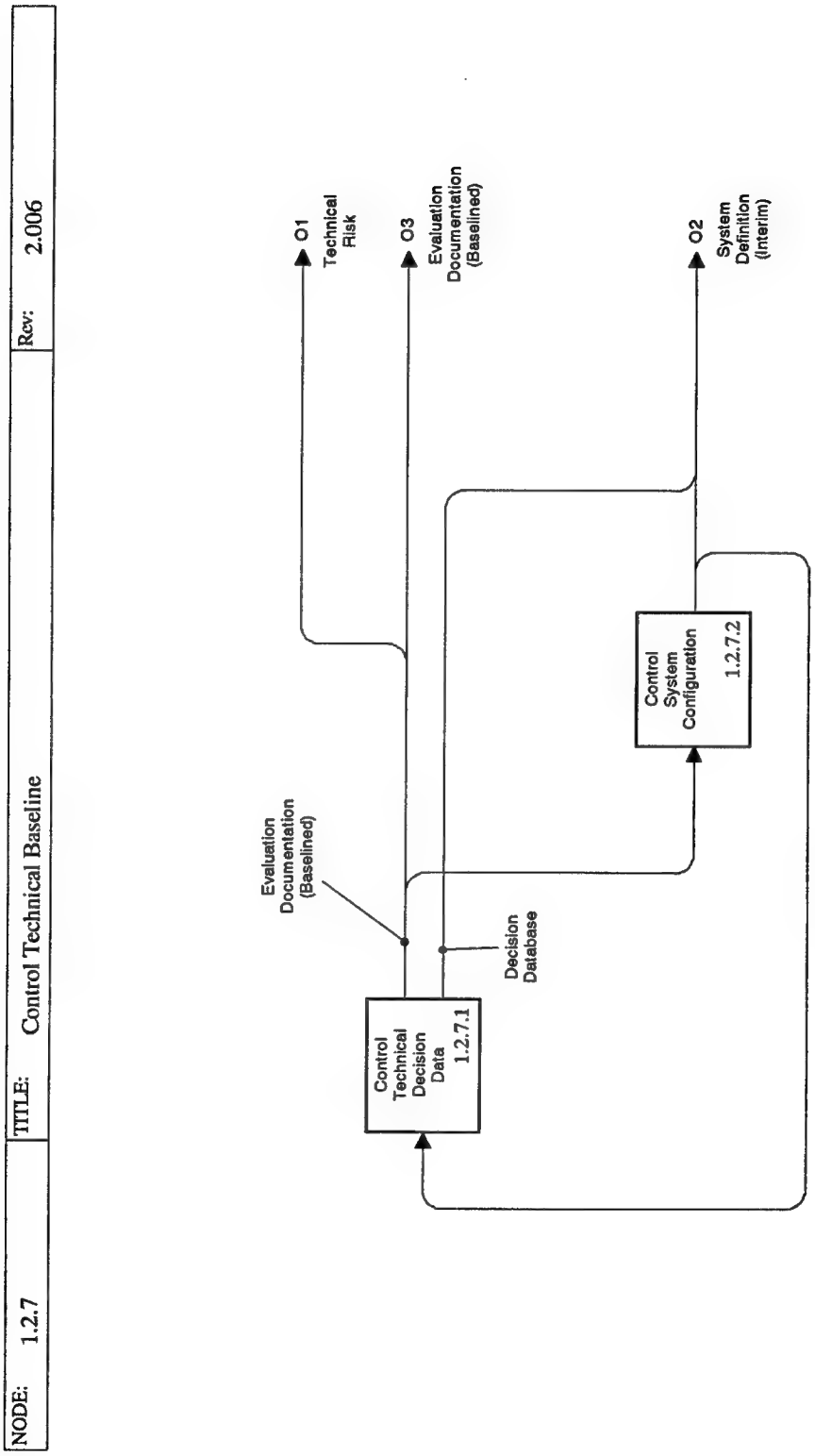


Figure 58. Control Technical Baseline

APPENDIX B. GSEP ACTIVITY DESCRIPTIONS

This appendix documents the GSEP activity descriptions, which describe the activity, list the entry and exit criteria, and define the activity inputs and outputs. The figure numbers in the activity descriptions refer to the diagrams that graphically depict the activities (if they exist) that are found in Appendix A. The entry and exit criteria define the partial ordering of the activities (i.e., what must be done before an activity can be initiated and what is required before an activity can be considered complete. The inputs and outputs to the activity (i.e., the work products) are defined in Appendix C.

In this appendix, inputs include IDEF inputs, controls, and mechanisms that indicate incoming flows. Outputs include IDEF outputs, controls, and mechanisms that indicate outgoing flows. The inputs and outputs listed do not include the tunneled inputs and outputs from higher level diagrams. However, entry and exit criteria may reference work products that are not explicitly listed as an input and/or output. The state of the work product that is input or output is identified by the <state> notation in the input and output lists.

Table 2 lists each of the GSEP activities, sorted by activity number, and the page number that contains a description of that activity.

Table 2. GSEP Activity Mapping for Appendix B

Table 2, continued

Activity Number	Activity Name	Page Number
1.1.3.3	Create Draft Increment Plan	Page 129
1.1.3.4	Commit to Plan	Page 132
1.1.4	Track Increment Development	Page 132
1.1.4.1	Monitor and Review	Page 133
1.1.4.2	Update Increment Plan	Page 134
1.1.4.3	Review Technical Product	Page 135
1.1.5	Develop System Plan	Page 136
1.1.5.1	Product Change Control	Page 136
1.1.5.2	Review Progress	Page 137
1.1.5.3	Update System Plan	Page 138
1.1.5.4	Commit to Proceed	Page 139
1.2	Define System Increment	Page 139
1.2.1	Analyze Needs	Page 141
1.2.1.1	Determine Stakeholders	Page 142
1.2.1.2	Assess Problems and Needs	Page 142
1.2.1.3	Define Environment	Page 143
1.2.1.4	Develop Informal Functionality	Page 144
1.2.2	Define Requirements	Page 145
1.2.2.1	Determine Behavioral Requirements	Page 146
1.2.2.2	Determine Performance Requirements	Page 147
1.2.2.3	Map Behavior to Performance	Page 148
1.2.2.4	Refine Requirements	Page 148
1.2.3	Define Functional Architecture	Page 149
1.2.3.1	Partition Requirements into Functions	Page 151
1.2.3.2	Define Lower Level Functions	Page 152
1.2.3.3	Define Partition/Functional Interfaces	Page 152
1.2.4	Synthesize Allocated Architecture	Page 153
1.2.4.1	Allocate Functional Architecture to Alternative Solutions	Page 154
1.2.4.2	Define Physical Parameters	Page 155
1.2.4.3	Define Physical Interfaces	Page 156
1.2.4.4	Integrate Design	Page 156
1.2.4.5	Define Physical Architecture	Page 157
1.2.5	Evaluate Alternatives	Page 157
1.2.5.1	Assess System	Page 159

Table 2, continued

Activity Number	Activity Name	Page Number
1.2.5.2	Identify and Assess Technical Risks and Problems	Page 160
1.2.5.3	Perform Sensitivity Analysis	Page 160
1.2.5.4	Allocate Performance to Technical Parameters	Page 161
1.2.5.5	Identify and Perform Trade-Offs	Page 162
1.2.5.6	Select System Solution	Page 162
1.2.6	Validate and Verify Solution	Page 163
1.2.6.1	Define Validation and Verification Test Procedures	Page 164
1.2.6.2	Validate System	Page 165
1.2.6.3	Verify System	Page 165
1.2.7	Control Technical Baseline	Page 166
1.2.7.1	Control Technical Decision Data	Page 167
1.2.7.2	Control System Configuration	Page 167

B.1 ENTERPRISE CONTEXT

Figure 42 (Appendix A) contains the decomposition of the Enterprise Context activity (−0).This activity defines the external context for the GSEP. Its purpose is to provide a frame of reference for the highest level GSEP activity, Develop Systems (0). The Manage Organization activity, although very important to the enterprise, is not part of the GSEP, and is not decomposed in this report.

Entrance Criteria N/A

Exit Criteria N/A

Inputs N/A

Outputs N/A

B.2 DEVELOP SYSTEMS

The Develop Systems activity (0) can be interpreted as the organizational endeavor to develop multiple systems, related or unrelated. Related systems may be technical spin-offs or similar systems that exist in a product family.

Figure 43 (Appendix A) contains the decomposition of the Develop Systems activity (0). This activity identifies the customer requirements, user requirements, and technology base used to develop the system. The high-level product and organizational plans (i.e., External System Definition, Subsystem/Component Status, and External System Plan) impact the way in which the system is developed. If the inputs and controls come from the level above (i.e., Above), they constrain the Develop System activity (1). If the inputs and controls come from the level below (i.e., Below), they inform and may result in the need for appropriate action in the Develop System activity (1). If the

inputs and controls come from across (i.e., Across), they either constrain or inform, as appropriate. The Develop System activity is responsible for developing all the systems that the organization will produce.

This activity also defines the interactions between higher level systems, subsystems, and components (see Section 2.3.2). Subsystems will receive information from the system or subsystem above and will send data to components or subsystems below. This activity also describes how information is communicated between peer activities using "across" information flows. Components will only receive data from systems or subsystems above them or subsystems/components across from them.

The interaction of subsystems or components with their peer subsystems or components is important for efficient communication of information and coordination of activities. This type of interaction is used to mitigate interface design issues and technical inconsistencies or problems in the system design. The GSEP supports this type of interaction. There are times when very tight interactions between somewhat disconnected subsystems are required; for example, when an operational component must be manufactured by a manufacturing component. The engineering team defining the operational system has to ensure that it can be manufactured, and visa versa. These interactions must be carefully monitored to ensure that system design changes do not adversely impact the design decisions that were made during the selection of the overall system design.

These interactions could be controlled by forcing all of the intersubsystem communications to formally go through the system above, but this would likely stifle the communication and beneficial creativity. On the other hand, it may be difficult for engineers with a subsystem- or component-level perspective to discern when a discussed/proposed change adversely affects the system design above. As a result, the engineers may not feel that the change warrants exposure to the upper level system, and the problems may not be found until later system integration.

Entrance Criteria	• The organization plan documents the structure for the system development. • A high-level product plan outlines the objectives for the system development. • Asset development goals are established. • The customer and user are identified.
Exit Criteria	• The system(s) is produced.
Inputs	• Customer Requirements • User Requirements • Technology Base • Organization Plan • Asset Development Goals • Product Plan
Outputs	• Systems

B.2.1 Develop System

The Develop System activity (1) includes the management, technical, and implementation subprocesses. For the purposes of the GSEP, Develop System is directly related to the "Development" phase of a system's life cycle. GSEP includes the decision processes required for optimizing and fully describing a system so that it can be fabricated and used in the manner in which it was intended. The GSEP does not include the development activities used to implement the system definition, but does include the interfaces to the implementation activities.

Figure 44 (Appendix A) contains the decomposition of the Develop System activity (1). This activity manages the system development effort, defines the system, and implements the system definition. The system may be the manufacturing, support, operational, etc. "subsystems." This activity includes both the management of the system development and the technical activities that produce the system.

The Implement System activity (1.3) produces the actual system by carrying out the System Definition that is defined in the GSEP process. No aspect of the system definition is to be modified in this activity. Instead, there must be iteration back into the GSEP activities (i.e., Manage Development Effort [1.1] and Define System Increment [1.2]) to make modifications to the System Definition. The activities that implement the system definition are not included in GSEP.

Entrance Criteria

- Channels to the user and customer are established.
- System objectives are established and documented in the organization plan.
- Technical constraints are documented in the external system definition.

Exit Criteria

- An approved System Definition (the system) is complete.

Inputs

- Customer Requirements
- External System Definition
- External System Plan
- Organization Plan
- Subsystem/Component Status
- Technology Base
- User Requirements

Outputs

- System
- System Definition < Approved >
- System Plan < Approved >
- System Status

B.2.1.1 Manage Development Effort

The Manage Development Effort activity (1.1) determines the context of the system development, analyzes risks in creating the system, develops an increment plan that defines a plan to reach a milestone in the development of the system definition, tracks the technical development of the system, and develops a system plan that manages the development of system definition for this system and any subsystems or components for which this system is responsible.

Figure 45 (Appendix A) contains the decomposition of the Manage Development Effort activity (1.1). This activity defines the management subprocess of the GSEP and includes activities related to the analysis of program risk, process tailoring, planning and scheduling, tracking the technical subprocess, and controlling and managing the entire systems engineering effort, potentially including the development of some or all of the constituent subsystems and components.

The Manage Development Effort activity is responsible for providing the delivery of the developed system. The system definition is always delivered through management rather than by the technical activities.

Entrance Criteria

- A signed contract or authorization to proceed with the system development phase of the life cycle exists.

Exit Criteria

- The system development phase of the life cycle is complete.

Inputs

- Customer Requirements
- Evaluation Documentation < Baselined >
- Evaluation Results
- External System Definition
- External System Plan
- Identified Problems and Needs
- Increment Status
- Organization Plan
- Subsystem/Component Status
- System Context

- System Definition < Interim >
- System Implementation Status
- Technical Baseline
- Technical Risk
- Technology Base
- User Requirements

Outputs

- Estimate of the Situation
- Increment Plan < Approved/Updated >
- System Definition < Approved >
- System Plan < Approved >
- System Status

B.2.1.1.1 Understand Context

The Understand Context activity (1.1.1) identifies and reviews all relevant information that could have an influence in the definition of the system. This activity is responsible for establishing what is to be pursued for the current development increment. This activity includes identifying the objectives, constraints, and alternatives that will be used in follow-on activities. Understanding Context includes defining the approach, estimating the situation, and reviewing the context definition.

Figure 46 (Appendix A) contains the decomposition of the Understand Context activity (1.1.1). This activity defines the context of the system being developed including all of the external constraints, the EoS, and review of the context definition.

Entrance Criteria

- A signed contract or authorization to proceed exists.
- Partial external system requirements or user requirements exist.

Exit Criteria

- The Estimate of the Situation is approved.

Inputs

- External System Definition
- Subsystem/Component Status
- System Context
- System Status
- Technical Baseline
- Technology Base
- User Requirements

Outputs

- Approach Definition

B.2.1.1.1.1 Define Approach

The Define Approach activity (1.1.1.1) defines key system/increment objectives, identifies system/increment constraints and stakeholders, and develops alternatives for meeting the system/increment objectives. This information is used by the Estimate Situation activity (1.1.1.2). The approach defined is based on the system context and the available resources.

The Define Approach activity (1.1.1.1) defines the objectives and identifies the alternatives, constraint, and stakeholders. Objectives must be results-oriented, clear and concise, controllable, measurable, reasonable, and appropriate for the current level of understanding about the system. Alternatives are different ways to meet the objectives. Constraints are limitations on alternatives. Stakeholders are individuals with a vested interest in the system being developed (e.g., customer, senior management, end-user). The objectives, alternatives, constraints, and stakeholders are used to establish a context for the system/increment.

Entrance Criteria

- A signed contract or authorization to proceed exists.
- Resources have been allocated to enact the increment activities.
- All stakeholders have approved updated system planning documents.
- Management has given authorization to proceed in current increment.

Exit Criteria

- Increment-level objectives have been defined and alternatives, constraints, and stakeholders have been identified to a level of detail that reflects the current level of understanding.
- The legacy inherited from previous increments has been analyzed, and appropriate changes have been made to the approach definition.

Inputs

- None

Outputs

- Approach Definition

B.2.1.1.1.2 Estimate Situation

The Estimate Situation activity (1.1.1.2) defines the EoS based on the approach definition generated by the Define Approach activity (1.1.1.1). The EoS documents assumptions, decisions and their rationale, and the approach definition.

The Estimate Situation activity (1.1.1.2) creates or updates the EoS. At a minimum, the EoS should identify and document the system approach defined in the Define Approach activity (1.1.1.1). The EoS may also document assumptions, decisions and their rationale, important historical information, and factors that may inhibit the successful development of the system. This information can be identified from sources like the contract and Statement of Work (SOW); insights about and from the client; historical project plans and budgets; client and business area policies, procedures, and standards; and interviews with key stakeholders at all levels.

Entrance Criteria

- The approach for the current increment has been defined.

Exit Criteria

- An Estimate of the Situation has been created/updated to a level of detail that reflects the current level of understanding.

- Identified stakeholders and problems and needs are understood and have been added to the Estimate of the Situation.

Inputs

- Approach Definition
- Estimate of the Situation < Draft >

Outputs

- Estimate of the Situation < Draft >

B.2.1.1.1.3 Review Context

The Review Context activity (1.1.1.3) takes the EoS and validates it against the stakeholder's expectations. This activity is also where the stakeholders commit to the system objectives and the approaches for meeting the objectives. This activity approves the EoS that is used in the risk assessment, the increment planning, and the development of the system plan.

The Review Context activity (1.1.1.3) obtains stakeholder consensus on the defined approach including what is to be developed this increment. Specifically, this activity ensures that:

- All stakeholders are identified and their expectations are clearly documented.
- Objectives are clearly documented and are reasonable.
- Current increment-level objectives are derived from or are refinements of any higher level system objectives.
- Proposed alternatives adequately address objectives and constraints.
- All stakeholders agree to pursue the system and current increment.

All key stakeholders review and approve the EoS. Although the EoS does not need to circulate widely outside the immediate development team, senior management should review, correct, and approve the EoS to ensure that the documented objectives are clear and correct.

The EoS may be modified as a result of reaching agreement. All changes are documented along with the rationale for making the change, and the final, approved EoS is distributed to all stakeholders.

Entrance Criteria

- The system Estimate of the Situation document has been drafted or updated to address the current increment.
- All individuals with an interest in the success of the system activities (i.e., stakeholders), or their representatives, are participating.

Exit Criteria

- The Estimate of the Situation is approved by the stakeholders and there is consensus to proceed.

Inputs

- Estimate of the Situation < Draft >

Outputs

- Estimate of the Situation < Approved >
- Estimate of the Situation < Draft >

B.2.1.1.2 Analyze Risk

The Analyze Risk activity (1.1.2) identifies potential long- and short-term risks. The process drivers identified in the EoS should be analyzed along with the risks to evaluate their impact on the

increment's development process. Mitigation strategies are developed for each significant risk. To ensure that all of the significant risks are identified, lessons learned from previous, similar programs are reviewed, and a risk questionnaire, if available, is utilized. All increment stakeholders should have input into this activity.

Figure 47 (Appendix A) contains the decomposition of the Analyze Risk activity (1.1.2). In this activity, risks are analyzed, risk aversion strategies are developed, and stakeholder commitment is made on aversion strategy execution.

Entrance Criteria

- Stakeholders, objectives, constraints, and alternatives, have been identified in an approved Estimate of the Situation.

Exit Criteria

- A Risk Management Plan has been created and approved.

Inputs

- Estimate of the Situation < Approved >
- External System Definition
- Subsystem/Component Status
- System Context
- System Status
- Technical Risk
- Technology Base
- User Requirements

Outputs

- Risk Management Plan < Approved >

B.2.1.1.2.1 Perform Risk Analysis

The Perform Risk Analysis activity (1.1.2.1) determines the likelihood and impact of the risks associated with the current increment of the system development. These risks are used to refine the process drivers as well as plan the risk aversion strategies.

The Perform Risk Analysis activity (1.1.2.1) is initiated by comprehensively identifying potential system and/or increment development risks. Objectives should be examined with respect to alternatives, constraints, and organizational and project assets, and potential risk areas should be identified. Unsatisfactory outcomes and the effect on both the current increment and system success criteria should be examined; that is, if a significant risk is identified for the current increment, it should be traced back to the system-level objectives and success criteria to determine whether the risk may affect the system as well as the increment. To help identify typical system risks, risk taxonomies such as in Boehm and Ross (1989, 117), U.S. Air Force (1988), and Charette (1990, 216–59) can be used.

Risks can be analyzed once they are identified by categorizing them and determining risk likelihood and consequence:

- Estimate the chance of potential loss (or gain) and the consequence (or benefit) of the risk situations previously identified.
- Analyze the risk dependencies (i.e., how one risk is impacted by another risk).
- Show any uncertainties in the risk likelihood and consequence estimates.

After completion of the risk analysis, a risk evaluation is performed to identify risk aversion strategies and examine the impact of the risk aversion strategy on each high risk item. Any risk aversion strategy identified should reduce the cost and/or probability of risk occurrence to an acceptable level.

It is possible for risk aversion strategies to introduce new risks that detract from the anticipated benefits or negatively affect other risks; therefore, it is important to assess the impact of a risk aversion strategy on other risks.

In general, consider the following when defining risk aversion strategies:

- Is the strategy feasible?
- Are resources critical to technical success identified?
- Does the strategy reduce risk to an acceptable level?
- Will the strategy negatively affect another risk?
- What is the potential impact of new risks, if any, introduced by the strategy?
- Does the strategy support increment and/or system development objectives?
- Are the tactics and means for implementing the strategy consistent with increment and/or system constraints?
- Is the strategy cost-effective?

The results of this activity should be documented in the draft RMP. The plan is enhanced and matured during the remainder of the Analyze Risk activities.

Using New Technologies: A Technology Transfer Guidebook (Software Productivity Consortium 1993c) can help the engineer identify and evaluate strategies that involve new technology. This guidebook provides detailed guidance on identifying and selecting new technologies and defines the process activities necessary to insert the technology for maximum use and benefit.

Entrance Criteria

- Stakeholders have reviewed and approved the Estimate of the Situation.
- Stakeholders have reached consensus to proceed with the current increment.

Exit Criteria

- The risks have been analyzed for the current increment. The identified risks include the probability and consequence of occurrence, the prioritization, and associated risk aversion strategies.

Inputs

- Identified Risks < Draft >

Outputs

- Identified Risks < Draft >

B.2.1.1.2.2 Review Risk Analysis

The Review Risk Analysis activity (1.1.2.2) is where the results of the perform risk analysis are assessed. At this point, the goal is to have the team look for any unaccounted risks or risks that are infeasible.

The Review Risk Analysis activity (1.1.2.2) is an opportunity for stakeholder review and comment on the results of the risk identification, analysis, and evaluation activities. In many cases, the most effective use of time is for the project manager and/or risk analyst to identify, analyze, and evaluate risks independently and then submit the results of the activities to the appropriate stakeholders for review. Review comments may identify the need to repeat some or all of the previous risk analysis activities.

Entrance Criteria

- All increment risks have been identified given the current level of understanding.
- The probability and cost of occurrence of each risk has been estimated.
- The highest priority risks, based on overall risk factors, have been determined.
- Risk aversion strategies for each high-priority risk have been identified, and the impact of the strategy on other risks and process drivers has been analyzed.
- All stakeholders have received the identified risks (draft) for review.

Exit Criteria

- The stakeholders reached consensus regarding any changes to the identified risks (draft).
- All agreements or changes to the identified risks (draft) have been documented and distributed to all stakeholders.

Inputs

- Identified Risks < Draft >

Outputs

- Identified Risks < Approved >
- Identified Risks < Draft >

B.2.1.1.2.3 Plan Risk Aversion

The Plan Risk Aversion activity (1.1.2.3) determines strategies for reducing the impact of risks in the system and increment development. The RMP is updated to include risk mitigation plans and the approach that will be used to monitor risks during the system and/or increment development.

In the Plan Risk Aversion activity (1.1.2.3), the high-level cost and schedule for each risk is estimated and evaluated, the best risk aversion strategies are determined for each. The risk aversion strategies are documented along with an RMP that documents the way the risk aversion strategies will be carried out. Ideally, the appropriate stakeholders are involved in this activity to help evaluate system effectiveness and recommend the best risk aversion strategies.

It is possible to perform risk reduction for extremely high risks through a special risk reduction increment. If a risk reduction increment is undertaken, it is important that any other system development that depends on the results of the risk reduction activities is not planned to start until after the risk reduction activities are complete.

The results of this activity should be documented in an update of the draft RMP.

Entrance Criteria

- An approved Risk Management Plan.

Exit Criteria

- The cost and schedule associated with each alternative and risk aversion strategy have been estimated.
- A recommended risk aversion for each risk item has been selected.

Inputs

- Identified Risks < Approved >
- Risk Aversion Strategy

Outputs

- Risk Management Plan < Draft >

B.2.1.1.2.4 Commit to Strategies

The Commit to Strategies activity (1.1.2.4) will validate the RMP. If the RMP is found lacking, it may be updated. This activity will also get the stakeholders to buy in to the RMP and tailoring strategies.

The Commit to Strategies activity (1.1.2.4) is a mechanism for formally informing all stakeholders of the contents of the RMP. During this activity, the effectiveness of the risk mitigation strategies and quality assurance plans is assessed. As a result, the consensus and commitment to the risk aversion strategies recommended for each development alternative and high-priority risk is reached. If consensus is not reached or commitment is not secured, or if the risk aversion strategies committed to are not the ones recommended, then the risk analysis activities may need to be repeated, as appropriate.

If modifications are made to the draft RMP, all changes and the rationale for the changes, are documented and distributed to all stakeholders.

Entrance Criteria

- The best candidate risk aversion strategy for each identified risk has been recommended.
- All individuals with an interest in the success of the system, or their representatives, have reviewed the draft Risk Management Plan and are participating.

Exit Criteria

- Stakeholders have reached consensus on a risk aversion strategy for each high-priority risk.
- Stakeholders have approved the updated Risk Management Plan.

Inputs
- Risk Management Plan < Draft >

Outputs
- Risk Aversion Strategy
- Risk Management Plan < Approved >

B.2.1.1.3 Plan Increment Development

The Plan Increment Development activity (1.1.3) uses the tailoring strategy and Risk Management Plan (RMP) to produce an increment plan. The increment plan defines the detailed plan for developing the increment and monitoring and reviewing the development. This activity reviews the significant increment risks and associated risk aversion activities and uses this information to create the increment process definition, plan, and schedule.

Figure 48 (Appendix A) contains the decomposition of the Plan Increment Development activity (1.1.3). This activity executes the risk aversion, reviews development alternatives for the increment being planned, creates the increment plan, and gets commitment to that plan.

Entrance Criteria
- The Risk Management Plan must be approved.

Exit Criteria
- The Increment Plan is approved.

Inputs
- External System Definition

Outputs
- Increment Plan < Approved >

B.2.1.1.3.1 Execute Risk Aversion

The Execute Risk Aversion activity (1.1.3.1) executes the risk aversions strategies that were planned in the Plan Risk Aversion activity (1.1.2.3). This might include things like prototyping, simulations, studies, or surveys. The RMP is updated to reflect the results of executing the risk aversion strategies.

The Execute Risk Aversion activity (1.1.3.1) executes the risk aversion strategies. These may include prototyping, simulation, surveys, comparative evaluations, evolutionary development, and other appropriate techniques. Complicated or long-term risk aversion activities may be performed in their own development increments. The results of the risk aversion activities impact the selection of development alternatives for this increment. The RMP is updated with all relevant risk aversion execution results.

Entrance Criteria
- Consensus on a risk aversion strategy for each high-priority risk item has been reached.

Exit Criteria
- The tasks supporting the risk aversion strategy have been performed.
- A development alternative has been determined and justified as a result of performing the risk aversion strategy.
- The effect of the development strategy on the increment and system process has been assessed.
- The Risk Management Plan has been updated, as appropriate.

Inputs

- Development Alternative
- Risk Management Plan < Approved >

Outputs

- Risk Management Plan < Updated >

B.2.1.1.3.2 Review Development Alternative

The Review Development Alternative activity (1.1.3.2) takes the results of the Execute Risk Aversion activity and uses these results to select a development alternative for the system increment being developed. The stakeholders are again part of the selection of the development alternative to ensure stakeholder commitment.

In the Review Development Alternative activity (1.1.3.2), the results of the Execute Risk Aversion activity are reviewed by appropriate stakeholders. Stakeholder commitment is solicited for the development alternative or approach that is selected based on the results of executing the risk aversion strategies. If the development alternative selected as a result of the risk aversion activities is not committed to by senior management, then additional risk aversion actions are required.

Entrance Criteria

- A development alternative has been determined and justified as a result of performing each risk aversion strategy.
- All individuals with an interest in the success of the system, or their representatives, are participating.

Exit Criteria

- Consensus has been reached on the development alternative or approach selected as a result of performing the risk aversion strategy.
- The updates to the Risk Management Plan have been approved.

Inputs

- External System Definition
- Risk Management Plan < Updated >

Outputs

- Development Alternative
- Risk Management Plan < Enacted >

B.2.1.1.3.3 Create Draft Increment Plan

The Create Draft Increment Plan activity (1.1.3.3) defines a plan, including schedule and resources, for developing the current system increment. During this activity, the activities, methods, and tools used to carry out the increment will be selected.

The main output of the Create Draft Increment Plan activity (1.1.3.3) is an enactable plan for developing the increment. The project's process definition must address increment objectives, development success criteria, constraints, development alternatives, risk mitigation strategies, and any other process driver.

In this activity, technical activities are identified, defined and ordered, and methods, practices, or tools for each task are specified. Activity definitions should include sufficient interfacing events (e.g., IPTs, management reviews, in-process reviews) to coordinate this increment's activities with other increments as well as to satisfy customer control and monitoring needs. Ongoing support activities, such as configuration control, quality assurance, and documentation, are also be included.

Although the basic management functions (planning, monitoring, and controlling) are performed continuously, resources for specific increment development and development support activities are identified, organized, and allocated after the increment risks are averted. The main output of the Create Draft Increment Plan activity (1.1.3.3) is an increment plan for the development of the product (or part of the product) that will be produced in the current increment, including a specification of the development process that will be used.

The increment plan can be documented as part of the system planning documents or it can be a standalone document. The plan should:

- Establish development goals and associated development success criteria that support the current increment objectives
- Estimate size and scope for the development to be accomplished in the current increment, including:
  - The number and type of product components (e.g., modules, documentation) to be produced as a result of enacting the increment plan
  - The size of each product component
  - The number, severity, and source of errors likely to be found during verification activities
- Identify the activities or methods to be performed in the current increment
- Define/redefine activities, methods, and supporting work products, if necessary
- Sequence the activities if not sequenced by the selected method
- Estimate development cost and schedule for each increment activity and allocate resources
- Select and allocate supporting tools
- Define work packages, or the lowest work breakdown structure (WBS) level, for the key activities defined for the current increment

In addition, the plan should:

- Address any customer requirements
- Include ongoing support activities in the increment plan, such as configuration management, quality assurance, and documentation
- Comply with customer policies, procedures, standards, and regulations, if necessary
- Comply with the organizational process definition, if available and appropriate
- Use any available historical planning and engineering data for estimating purposes
- Take advantage of organizational methods, tools, training, and support

Entrance Criteria

- The Risk Management Plan must be approved.
- Commitment to the development alternative has been secured.

- System status and engineering data from previous increments has been collected and analyzed, and the system planning documents have been updated accordingly.

Exit Criteria

- Increment activities have been instantiated in the increment plan.
  - The increment activities are consistent with the objectives of the increment and satisfy the increment success criteria as defined in the current version of the Estimate of the Situation.
  - The increment activities are consistent with the development alternative as documented in the current version of the Risk Management Plan.
  - The increment activities are consistent with the organizational process definition, if any, or if appropriate.
  - Increment activities have been defined or tailored from the organizational process definition, if any, as required to meet the increment objectives and success criteria, accommodate increment alternatives, satisfy increment constraints, and avert increment risks.
  - The dependencies between activities and the dependencies of the tasks within each activity have been documented.
  - Costs have been estimated, and work packages or the lowest WBS level has been opened for the key increment activities.
  - Durations have been estimated for each activity.
  - Resources have been allocated to each activity.
  - Intermediate milestones and performance measurements have been defined for each increment activity.
  - The Monitoring and Review activities have been scheduled to accumulate data for and analyze the status of increment activities.
  - Sufficient interfacing events have been scheduled, such as interface control working groups, management reviews, and in-process reviews, to coordinate this increment's activities with other increments.

Inputs

- External System Definition
- Increment Plan < Draft >
- Risk Management Plan < Enacted >

Outputs

- Increment Plan < Draft >

B.2.1.1.3.4 Commit to Plan

The Commit to Plan activity (1.1.3.4) validates the increment plan. If the plan requires modifications, it is returned to the Create Draft Increment Plan activity(1.1.3.3). Ensuring stakeholder commitment and buy-in is an important part of the Commit to Plan activity.

The Commit to Plan activity (1.1.3.4) is an opportunity for stakeholders to review and comment on the results of the increment planning and scheduling activities. Consensus that the activities of the increment plan are appropriate to meeting increment objectives must be reached. The increment plan may be updated according to stakeholders' comments. When commitment to the detailed increment plan is secured, a formal briefing to senior management on the current increment development activities can occur. This briefing may be a natural extension of any existing, periodic internal management reviews, or it may be scheduled separately. As a result of committing to the increment plan, work packages are opened and assigned and the detailed increment process, as documented in the increment plan, is considered enactable.

Entrance Criteria

- A draft Increment Plan has been instantiated to an enactable level of detail.

Exit Criteria

- Consensus has been reached regarding any changes to the draft Increment Plan.
- All agreements or changes in the meeting minutes have been documented and distributed to all stakeholders.
- Consensus has been reached to proceed with the increment by enacting the Increment Plan.
- Work packages have been opened.

Inputs

- Increment Plan < Draft >

Outputs

- Increment Plan < Approved >
- Increment Plan < Draft >

B.2.1.1.4 Track Increment Development

The Track Increment Development activity (1.1.4) is the main management interface into the technical activities. This activity uses metrics provided by the technical systems engineering activities to assess progress of the technical activities. Minor modifications to the increment plan created in the Plan Increment Development activity (1.1.3) are permitted, but if major replanning is necessary, this activity is terminated and the Develop System Plan activity (1.1.5) is initiated. This activity includes a review of the technical product being produced during this increment. This review includes a comparison of the completed work products with the increment's success criteria. All officially delivered system definitions are distributed through this activity.

Figure 49 (Appendix A) contains the decomposition of the Track Increment Development (1.1.4) activity. This activity monitors and reviews the progress of the increment, makes minor updates to the increment plan, if necessary, and reviews the system increment work products being produced.

Entrance Criteria

- The Increment Plan must be instantiated to an enactable level of detail, and it must be approved.

Exit Criteria

- The technical development activities for the increment must be complete, and the technical product must be reviewed.

Inputs

- External System Definition
- Increment Plan < Approved >
- Increment Status
- Risk Management Plan < Approved >
- Technical Baseline
- Technical Risk
- Technology Base
- User Requirements

Outputs

- Increment Plan < Enacted >
- Increment Plan < Updated >
- Increment Status Reports
- System Definition < Approved >

B.2.1.1.4.1 Monitor and Review

The Monitor and Review activity (1.1.4.1) assesses the progress of the increment development.

The Monitor and Review activity (1.1.4.1) maintains management control over the technical development and ensures that the technical activities are carried out in accordance with the management plan. This activity periodically captures and analyzes the status of the increment to provide management with insight into the cost, schedule, process fidelity and fitness, and technical performance status of the increment. Raw activity progress/status is analyzed to produce management metrics that support decision making regarding corrective action. Work package status for each cost account is collected, analyzed, and summarized to the highest schedule level. For each parameter selected for monitoring, actual and projected performance is updated, trends are analyzed, and unfavorable trends or values are noted. The following should be reviewed and analyzed based on the data and measurements collected during the enactment of the technical activities that make up Define System Increment (see Figure 44, page 100):

- Product component size status
- Product component error data
- Product component change data
- Increment schedule status

- Increment cost status
- Increment risk status
- Increment quality assurance status

Periodic increment reviews are conducted as a part of this activity. These reviews serve two purposes. First, they demonstrate to the team and other stakeholders that the current increment and overall system development are under control. Second, they provide a mechanism to obtain a commitment to proceed with the increment plan. At these reviews, decisions can be made to reallocate resources, replan schedules, or reassess risks for the current increment activities. All such changes are reflected in an updated increment plan and/or evolved increment process.

The *Software Measurement Guidebook* (Software Productivity Consortium 1992) provides information on specific size, cost, schedule, and error measurement and analysis methods.

Entrance Criteria

- The increment process is being enacted as defined in the Increment Plan.

Exit Criteria

- Product component and increment data has been collected and reviewed.
- Status reports have been produced and distributed.

Inputs

- Increment Plan < Approved >
- Increment Plan < Updated >
- Increment Status
- Increment Status Reports
- System Definition < Interim >
- Technical Baseline

Outputs

- Increment Plan < Enacted >
- Measurements and Status
- System Definition < Interim >
- Technical Status

B.2.1.1.4.2 Update Increment Plan

The Update Increment Plan activity (1.1.4.2) uses the assessed progress that was calculated during the Monitor and Review activity (1.1.4.1) to update the increment plan.

The Update Increment Plan activity (1.1.4.2) makes minor modifications (i.e., no perceived risk) to the increment plan to account for technical and managerial problems and situations. For example, if a particular tool is unavailable when scheduled, this activity may decide to either postpone the task

that requires the tool or substitute another tool. If major modifications (i.e., perceived risk) to the plan are necessary, then the Develop System Plan activity (1.1.5) is initiated so that all impacts of the required modifications to the increment development can be assessed relative to the entire system development effort.

Entrance Criteria

- Measurement and Status information is produced as a result of minor discrepancies between the Increment Plan and Increment Status.

Exit Criteria

- An updated Increment Plan has been produced.

Inputs

- Measurements and Status

Outputs

- Increment Plan < Updated >

B.2.1.1.4.3 Review Technical Product

The Review Technical Product activity (1.1.4.3) evaluates the system definition against the increment success criteria. If the system definition requires modification, the updates are again monitored and reviewed in the Monitor and Review activity (1.1.4.1). If the system definition remains incorrect or incomplete, then a new increment may be necessary. In this case, the Develop System Plan (1.1.5) is initiated, lessons learned from this increment are documented, and planning for the next increment proceeds.

During the Review Technical Product activity (1.1.4.3) a technical product review is performed. The technical product review is a stakeholder review of the product (or part of the product) developed in the current increment. The purpose of this activity is to ensure that increment objectives and development success criteria have been met. This review is an opportunity for stakeholder review; the ongoing Monitor and Review activity, however, is responsible for identifying and planning to resolve any problems with meeting increment objectives or development success criteria. The Increment Plan is only updated here when risk analysis is deemed unnecessary. If it is not clear that risk analysis is unnecessary, then the management team should cycle through the Manage Development Effort activities (1.1) (see Figure 45, Appendix A).

Entrance Criteria

- The increment process, as defined in the Increment Plan, has been enacted and has produced a product or part of a product.

Exit Criteria

- The product or part of the product has been reviewed to ensure that increment objectives and success criteria have been met.
- The results of verification activities have been reviewed to ensure the technical quality of the product (or part of the product).
- Unique configuration identification has been assigned to the product (or part of the product) approved for placement under configuration control.

Inputs

- System Definition Interim
- Technical Baseline
- Technical Status

Outputs

- Increment Status Reports
- System Definition < Approved >
- System Definition < Interim >

B.2.1.1.5 Develop System Plan

The Develop System Plan activity (1.1.5) defines the long-term plan for the development of foreseeable system increments, including the planning constraints for constituent subsystems and components. This activity coordinates the planning and statusing of the system as well as its subsystems and components.

Figure 50 (Appendix A) contains the decomposition of the Develop System Plan activity (1.1.5). This activity implements product change control, reviews progress of the technical development effort, updates the long-term system plan for the foreseeable future, and gets commitment to proceed with the updated system plan.

Entrance Criteria

- The current increment status must be understood.
- The Risk Management Plan is approved.

Exit Criteria

- A System Plan has been approved.

Inputs

- External System Definition
- Increment Plan < Enacted >
- Increment Status Reports
- Risk Management Plan < Approved >
- Subsystem/Component Status
- System Definition < Approved >
- System Implementation Status

Outputs

- System Plan < Approved >
- System Status

B.2.1.1.5.1 Product Change Control

The Product Change Control activity (1.1.5.1) baselines any externally deliverable system definition created in this increment.

During the Product Change Control activity (1.1.5.1) the product (or part of the product) resulting from any externally deliverable increment is baselined. The implementation of changes is tracked to control products to ensure that the configuration of the product is controlled at all times. Each

baseline is established by identifying the configuration units that constitute the baseline. All subsequent changes are tracked in the configuration status. The history of changes to each configuration item is maintained throughout the system development. All items are stored under control in a secure, limited-access repository.

Entrance Criteria

- A configuration management plan is included as one of the system planning documents.
- A product (or part of the product) has been developed as a result of enacting the increment process.
- A configuration identification has been received.

Exit Criteria

- The baseline for the current increment product (or part of the product) has been established.

Inputs

- External System Definition
- System Definition < Approved >

Outputs

- System Baseline

B.2.1.1.5.2 Review Progress

The Review Progress activity (1.1.5.2) assesses the progress of the system increment (e.g., compares the increment's system definition against the increment's development success criteria) including the progress of any of this system's constituent subsystems or components. The system status is also collected, and the appropriate information is summarized to update the system metrics and identify additional process drivers.

In the Review Progress activity (1.1.5.2), actual measures are evaluated against those estimated in the increment plan, development success criteria are examined to ensure that they were met, and lessons learned are identified. An analysis of how the final increment status may affect the process for the rest of the system development is conducted. Specifically, increment data is reviewed and verified against the actual measurements.

Entrance Criteria

- The increment process, as defined in the Increment Plan, has been enacted.
- A baselined product for the system exists.

Exit Criteria

- Final increment status has been reviewed.
- Actual-versus-estimate metrics for both the increment and the system have been produced.
- Any lessons learned from the increment have been compiled.
- Changes to system development process drivers have been identified based on the enacted increment process and resulting increment product, data, and status.

Inputs

- Increment Plan < Enacted >
- Increment Status Reports
- Subsystem/Component Status
- System Baseline
- System Implementation Status

Outputs

- Metrics and Process Drivers < Updated >
- System Status

B.2.1.1.5.3 Update System Plan

The Update System Plan activity (1.1.5.3) uses the metrics and process drivers that result from the progress review of this system increment, as well as the status of the higher level system of which this system is a part (if applicable), to update the system plan.

During the Update System Plan activity (1.1.5.3), the system planning documents are updated to reflect the lessons learned from the increment. The main output of the Update System Plan activity is the updated system plan. The updated system plan includes new process drivers that were identified during the analysis of the increment's development progress. The system plan is also updated to reflect the current understanding of system objectives, constraints, development alternatives, and risk mitigation strategies, and to incorporate the legacy inherited from all enacted increments.

Entrance Criteria

- Increment progress has been reviewed, and all system and increment metrics and assumptions, as documented in the Estimate of the Situation, have been updated.

Exit Criteria

- The System Plan has been updated to reflect the all relevant lessons learned from the increment.

Inputs

- Increment Plan < Enacted >
- Increment Status Reports
- Metrics and Process Drivers < Updated >
- Subsystem/Component Status
- System Plan < Updated >

Outputs

- System Plan < Updated >

B.2.1.1.5.4 Commit to Proceed

The Commit to Proceed activity (1.1.5.4) will validate the system plan. If the plan is found lacking, it may be updated. This activity will also get the stakeholders to buy in to the system plan.

A key concept during the Commit to Proceed activity (1.1.5.4) is the commit concept, which comes into play at the end of the increment. The commit concept is one of the most important in using the GSEP management activities. It is similar in purpose to a baseline; all stakeholders are briefed on the results of the current increment and on any changes in system development plans and agree with the decisions made regarding what to do next. Stakeholders include project management at all levels and the IPTs. The review may also include customer representatives. The purpose of the review is to determine whether system-level objectives, alternatives, and constraints are still feasible, to agree on the objectives for the next increment, and to commit resources to that increment. The system planning documents may be modified as the stakeholders reach agreement. All changes should be documented with the change rationale and distributed to all stakeholders.

Entrance Criteria

- The system planning documents are drafted for the entire system development effort or updated following an enacted increment.
- All individuals with an interest in the success of the system, or their representatives, are participating.

Exit Criteria

- Consensus to proceed has been reached as a result of reviewing the system planning documents.
- Consensus on the objectives and success criteria for the next increment has been reached.
- All agreements or changes to the system planning documents have been documented and have been distributed to all stakeholders.

Inputs

- System Plan Updated

Outputs

- System Plan < Approved >
- System Plan < Updated >

B.2.1.2 Define System Increment

The Define System Increment activity (1.2) is the technical subactivity of the GSEP and is responsible for transforming a set of needs into a complete, balanced, system solution. All the GSEP technical activities are subactivities to this activity. This activity determines user and customer needs, defines systems requirements, defines a functional architecture, synthesizes the allocated architecture, selects a life-cycle optimized solution, and validates and verifies the system solution. This activity produces a verified and validated system design that is used to specify subsystems or components, or it may be directly implemented. When subsystems and/or component definitions are completed, the "as built" definitions and evaluation results are passed back up to the Develop System activity (1) and used to update and evaluate the system design.

Figure 51 (Appendix A) contains the decomposition of the Define System Increment activity (1.2). This activity is the technical subactivity in the GSEP that analyzes the needs the system must address, defines the system requirements, defines the functional architecture that provides an allocation-independent design that meets the system requirements, synthesizes allocated architectures that meet the system requirements, evaluates each of the alternatives and selects the

most desirable, validates and verifies the technical work products, and controls the technical baseline.

The Evaluate Alternatives (1.2.5) and the Validate and Verify Solution (1.2.6) activities are tightly coupled. The Evaluate Alternatives activity is responsible for the optimization of various systems designs and the trade-off of those designs against each other. During the Validate and Verify Solution activities, the intent is to ensure that the optimized work products do what they are supposed to do. In verification, what the verified work product supposed to do is interpreted by the preceding activity(s) work products. In validation, what the design is supposed to do is interpreted by the user/customer.

Entrance Criteria

- An approved or updated increment plan exists.
- If this increment is a subsystem or component, then the higher level external system definition must exist.
- Customer/user input must be available.

Exit Criteria

- A technical baseline for an increment of the system definition exists.
- The increment status is defined.
- The technical risks and problems associated with the increment are identified and documented.

Inputs

- Customer Requirements
- Estimate of the Situation
- Estimate of the Situation < Approved >
- External System Definition
- Increment Plan < Approved >
- Increment Plan < Approved/Updated >
- Increment Plan < Updated >
- System Definition < Approved >
- System Definition/Process Requirements
- Technology Base
- User Requirements

Outputs

- Increment Status
- System Context
- Technical Baseline
- Technical Risk

B.2.1.2.1 Analyze Needs

The Analyze Needs activity (1.2.1) determines the stakeholders, assesses the problems the system is to solve and the needs that the system is to address, defines the environment in which the system is to operate, and develops the informal functions the system is to perform. The goal of Analyze Needs activity is to define the true needs of the users and customers by fully understanding the solution stakeholders (who they are, what they really need, and why they need it) and identifying the potential environments in which a system solution will be manufactured, operated, and maintained.

Figure 52 (Appendix A) contains the decomposition of the Analyze Needs activity (1.2.1). This activity determines the stakeholders, assesses the problems the system is to solve and the needs that the system is to address, defines the environment in which the system is to operate, and develops the informal functions the system is to perform.

Entrance Criteria

- Configuration control constraints are defined.
- Existing technologies have been identified and are understood.
- User requirements are available. Note that this includes direct interaction with the user, if possible.

Exit Criteria

- The stakeholders are identified, and a strategy is developed to obtain expectations from a representative subset of the stakeholders.
- Operational scenarios are established.
- The system environment(s) is identified. This environment (i.e., where the system is to be used) is bounded or characterized in a way that allows engineering evaluation of the applicability/performance of the system.
- Informal functional hierarchy is defined. The informal functional hierarchy defines the high-level functions that must be performed in order to satisfy the needs and problems the system must address. These functions must be complete for the increment of the system being created.

Inputs

- Configuration Control/Constraints
- Estimate of the Situation
- External System Definition
- System Definition/Process Requirements
- System Requirements
- Technology Base
- User Requirements
- Validation & Verification Results

Outputs

- Informal Functional Hierarchy
- System Context
- User Specification

B.2.1.2.1.1 Determine Stakeholders

The Determine Stakeholders activity (1.2.1.1) identities the system stakeholders. These stakeholders are consulted throughout the definition of the system and, especially, at key decision points. The system stakeholders identified and consulted in this activity are not identical to those identified in the management subprocess. These stakeholders will overlap, but there may be unique stakeholders in each group. As an example, a stakeholder may be identified in the user community for representing the users in the management subprocess, but in the technical subprocess, several users that represent specialists may be identified.

During the Determine Stakeholders activity (1.2.1.1), the system stakeholders are identified based on preliminary needs for the system and its anticipated life cycle. This includes the identification of all elements that are involved in system definition, fabrication, and use (operations and support).

Entrance Criteria

- The user and customer requirements must exist and be available.

Exit Criteria

- The stakeholders are identified, and a strategy is developed to obtain expectations from a representative subset of the stakeholders.

Inputs

- Estimate of the Situation
- Identified Problems and Needs
- Informal Functional Hierarchy
- System Requirements

Outputs

- Identified Stakeholders
- System Context

B.2.1.2.1.2 Assess Problems and Needs

The Assess Problems and Needs activity (1.2.1.2) identifies the problems the system is being created to solve as well as the needs the system is to meet. During this activity, these problems and needs must be classified as to importance. Conflicts between needs and problems must be resolved. Measures of effectiveness and critical influencing factors must be identified and the operational concept defined. The identified stakeholders are used to help identify these needs and problems.

The Assess Problems and Needs activity (1.2.1.2) completely analyzes the user and customer concerns that a system solution will be designed to solve and the operational scenarios that support the system operational concept. Problems and needs must be viewed in terms of "potentials," in that many of the problems will arise in the future or unanticipated needs will arise during system development or use.

Measures of effectiveness are defined for assessing the ability of the system to address the identified problems and needs.

The domain of the problem being considered is often larger than the initial stated needs of the user and the customer. The goal of this activity is to identify user and customer problems and needs, known and anticipated, at their most basic level and not strictly a user/customer interpretation of them. Another objective during this activity is to define the critical influencing factors that drive the technical development of the system.

Entrance Criteria

- The stakeholders are identified, and some mechanism for interaction with them is established.
- The Estimate of the Situation is available.

Exit Criteria

- The problems and needs that the system will address are identified and captured. There must be some level of buy-in from the stakeholders. At a minimum, the customer and user representatives must be satisfied.
- The Estimate of the Situation is factored into the assessment of the problems and needs.
- Operational scenarios are established.

Inputs

- Estimate of the Situation
- Identified Environments
- Identified Stakeholders
- Informal Functional Hierarchy

Outputs

- Identified Problems and Needs

B.2.1.2.1.3 Define Environment

The Define Environment activity (1.2.1.3) identifies the environment in which the system must be manufactured, operated, and supported.

During the Define Environment activity (1.2.1.3), the system operational environment is determined. This determination must be based on the problems and needs that have been identified. The environment defines the context of all possible operational scenarios for the system throughout its life cycle.

Entrance Criteria

- The problems and needs are available.

Exit Criteria

- The system environment(s) is identified. This environment (i.e., where the system is to be used) is bounded or characterized in a way that allows engineering evaluation of the applicability/performance of the system.

Inputs
- Identified Problems and Needs
- Informal Functional Hierarchy

Outputs
- Identified Environments

B.2.1.2.1.4 Develop Informal Functionality

The Develop Informal Functionality activity (1.2.1.4) defines an informal functional architecture that defines the system functions at a very high level. These functions should in no way limit or constrain the solution space of the system alternatives. The purpose of these functions is to constrain and bound the problem domain, not to define the anticipated solution.

During the Develop Informal Functionality activity (1.2.1.4), the user needs are decomposed into high-level informal functions that allow restatement of the needs in engineering terms. These functions are very loosely defined but capture the intent of the user needs. Technical performance measures are developed that can be used to determine whether the functions adequately meet the users needs.

Entrance Criteria
- The environment(s) in which the system operates is defined.
- The problems and needs are defined.
- Technical performance measures are established and captured.

Exit Criteria
- A complete set of informal functions that satisfy the user and customer needs is defined and documented.
- The high-level interactions between the informal functions are identified.
- The customer and user agree that the functions identified will satisfy their needs.

Inputs

Identified Environments
- Identified Problems and Needs

Outputs
- Informal Functional Hierarchy

B.2.1.2.2 Define Requirements

The Define Requirements activity (1.2.2) takes the informal functional hierarchy produced by the needs analysis and defines the requirements that the system must satisfy to be acceptable to the user and customer of the system. The requirements are used to stipulate the conditions (predicted, or legally binding) under which the system will be purchased/procured by the customer.

Figure 53 (Appendix A) contains the decomposition of the Define Requirements activity (1.2.2). This activity takes the informal functional hierarchy and the user specifications produced by the needs analysis and, in conjunction with direct interactions with the customer and user, identifies the

requirements that ensure the system will meet the customer and users needs. The technical performance measures defined in the informal functional hierarchy are refined and augmented, and critical performance measures identified. Critical performance measures will be tracked closely in the evaluation, validation, and verification of the system. Behavioral requirements are defined that ensure that the system will generate appropriate responses to system input or temporal events.

Performance requirements are derived from the needs and assessed against their effect on the system's ability to meet the customer/user requirements. These performance requirements include physical requirements (e.g., weight, power consumption) as well as other quantitative requirements (e.g., timing, reliability, maintainability) that define the performance characteristics of the behavioral requirements.

After the behavioral and performance requirements are established, they are merged to define the system requirements by identifying the behavioral requirements (if any) that the performance requirements constrain.

Finally, the requirements are refined by analyzing the consistency and completeness of the requirements as well as their effectiveness in guaranteeing the satisfaction of the customer and user needs.

Entrance Criteria

- The informal functional hierarchy is available.
- The user and customer requirements are available.
- The environment(s) in which the system will operate is identified.
- A complete and consistent set of requirements that define both behavioral and performance requirements is defined. These requirements must define the behavior that is required to satisfy the informal functions and constrain the behavior via performance requirements that specify the quality of the behavior. The quality of the behavior is described in terms of technical performance objectives that are measurable.
- The user and customer agree that, if the defined requirements are met, the system will satisfy the identified problems and needs.

Inputs

- Configuration Control/Constraints
- Decision Database
- Derived Requirements
- Identified Environments
- Identified Problems and Needs
- Informal Functional Hierarchy

- System Definition < Interim >
- System Definition/Process Requirements
- Technology Base
- User Requirements
- User Specification
- Validation & Verification Results

Outputs

- Performance Requirements
- System Definition < Interim Updated >
- System Requirements

B.2.1.2.2.1 Determine Behavioral Requirements

The Determine Behavioral Requirements activity (1.2.2.1) formalizes the informal functional hierarchy into a set of testable behavioral requirements that define the behavior of the system that is required to produce the mandatory system outputs.

The Determine Behavioral Requirements activity (1.2.2.1) starts with an initial set of functions, generated from the customer and user inputs, and determines the behavioral requirements for the system. These requirements must be spawned from each of the identified informal functions and should not dictate a design solution. During the identification of the behavioral requirements, the technical performances measures are refined, augmented, and associated with the relevant behavioral requirements. The behavioral requirements do not specify performance requirements such as timing performance or weight. These requirements establish the data requirements that specify the information the system is to produce.

Entrance Criteria

- The informal functional hierarchy is available.
- The user requirements are available.
- The user specification is available.

Exit Criteria

- Behavioral requirements are defined that guarantee the production of all outputs the system is required to produce. These behavioral requirements specify all of the informal functionality that the system must satisfy.
- The customer and user agree that a system that complies with these requirements will satisfy the identified problems and needs.

Inputs

- Derived Requirements
- Informal Functional Hierarchy

- System Definition/Process Requirements
- System Requirements Interim
- User Specification

Outputs

- Behavioral Requirements

B.2.1.2.2.2 Determine Performance Requirements

The Determine Performance Requirements activity (1.2.2.2) formalizes the quantitative aspects of the informal functional hierarchy. The performance requirements define a set of testable performance objectives that define the quantitative aspects of the behaviors the system is to perform. These quantitative aspects define physical characteristics and define the performance characteristics of the behavioral requirements.

During the Determine Performance Requirements activity (1.2.2.2), the user needs are assessed and performance characteristics that are mandated in order to meet those needs are defined. These performance requirements are used to constrain the implementation of the behavioral requirements and the physical system characteristics. The performance requirements include requirements that define the required physical characteristics of the system, such as weight, as well as characteristics of the behavior that define how well the behavior must be carried out. All of the characteristics must be quantifiable and testable. The characteristics include things such as accuracy of results, the efficiency of the system, or the speed with which the system must accomplish the behavior. During the identification of the performance requirements, the technical performances measures are refined, augmented, and associated with the relevant performance requirements.

Entrance Criteria

- The informal functional hierarchy is available.
- The user specification is available.
- The behavioral requirements are available (optional, but recommended).

Exit Criteria

- The performance requirements are defined. The performance requirements qualify the way in which the behavioral requirements must be satisfied.
- Constraints are identified that define the range of performance that is acceptable for the system.
- The customer and user agree that a system that complies with these requirements will satisfy the identified problems and needs.

Inputs

- Behavioral Requirements
- Derived Requirements
- Informal Functional Hierarchy

- System Definition/Process Requirements
- System Requirements < Interim >
- User Specification

Outputs

- Performance Requirements

B.2.1.2.2.3 Map Behavior to Performance

The Map Behavior to Performance activity (1.2.2.3) correlates the performance requirements to the behavioral requirements that are affected by the performance requirements.

During the Map Behavior to Performance activity (1.2.2.3) the performance requirements are mapped to the behavioral requirements they constrain. Often, the performance requirements are related to particular behaviors in the system. Other times, the performance requirements are written against the system as a whole and will not map to any individual behavior.

Entrance Criteria

- Behavioral and performance requirements exist.

Exit Criteria

- Performance requirements are mapped to the behavioral requirements they affect.
- Any constraints on the performance requirements are related to the set of behavioral requirements that they affect. At this point, the performance requirements must not be subsetted to the behavioral requirements.

Inputs

- Behavioral Requirements
- Performance Requirements
- System Requirements
- Interim

Outputs

- Behavioral/Performance Requirements Mapping

B.2.1.2.2.4 Refine Requirements

The Refine Requirements activity (1.2.2.4) refines the requirements to address derived requirements discovered during the association of the performance and behavioral requirements. This activity also ensures that the system requirements are testable.

After initial requirements are determined, the Refine Requirements activity (1.2.2.4) evaluates them to ensure that they map to identified functions and that the requirements are testable. Once established, refined, and baselined, these requirements serve as the foundation upon which the system is defined. The Refine Requirements activity interacts with other technical activities that identify derived requirements. These derived requirements must pass through refinement before they are accepted and baselined. The activity must also ensure that the "derived" requirements are necessary

and sufficient to meet the objectives of the original requirements. The customer must confirm that the requirements satisfy their needs and expectations. Derived requirements are not based on design decisions; instead they are based on implications of other requirements.

In this activity, the requirements are also assessed against their impact on the system's measure of effectiveness. The sensitivity of the performance requirements to the resultant system's measure of effectiveness is assessed, and the performance requirements may be refined based on this sensitivity.

Entrance Criteria

- A set of requirements that define an integrated set of performance and behavioral requirements is defined.
- The user specification is available.

Exit Criteria

- A complete and consistent set of requirements that define both behavioral and performance requirements are defined. These requirements define the behavior that is required to satisfy the informal functions and constrain the behavior via performance requirements that specify the quality of the behavior. The quality of the behavior is described in terms of technical performance objectives that are measurable.
- The user and customer agree that if the requirements are met, the system will satisfy the identified problems and needs

Inputs

- Behavioral/Performance Requirements Mapping
- Derived Requirements
- User Specification

Outputs

- System Requirements
- System Requirements < Interim >

B.2.1.2.3 Define Functional Architecture

The Define Functional Architecture activity (1.2.3) creates a functional architecture made up of functions and interfaces. In this activity, an allocation-independent design of the system is created. The top-level functions are defined by partitioning the system requirements. These functions must meet allocation constraints that must be considered in the creation of the system (e.g., political considerations [subsystems that need to be developed in appropriate states], family of hardware configurations [existing hardware that is likely to be used for system implementation]).

Figure 54 (Appendix A) contains the decomposition of the Define Functional Architecture activity (1.2.3). This activity defines functions based on a partitioning of the system requirements. The partitioning of the system is based on partitioning criteria that include performance and design considerations. These functions are based on design decisions that begin to focus on a solution. As such, the functions are now constrained to be those that the systems engineer believes will result in feasible solutions based on reasonable implementation environments. This activity also determines the

interfaces, in terms of data structures, that must exist between the functions in order to meet the required system behaviors.

As the system functions are identified, alternate solutions that meet the requirements are identified. These solutions often demand a lower level decomposition of the system functions that define the intended solution. These lower level functions are identified and defined in the functional architecture. In some cases, interfaces are not cleanly defined. This may require a repartitioning of functions or functional overlap, where the same function appears to be occurring in several areas or at various levels of system decomposition.

The development of the alternative functional architectures does not result from a decomposition of the informal functions. Instead, the alternative functional architectures are based on the specified system requirements. The functions are not constrained by the informal functions defined earlier. This distinction is made to ensure that early assumptions and biases do not influence the system design. As an example, assume that a navigation function is an informal function in the system. It may be that there are multiple lower navigation functions that are better combined with vehicle control functions and other navigation functions that are better combined with communication functions. This may result in an autopilot function and a communication function that are not a decomposition of the navigation functions but functions that were defined from analysis of the system-level requirements instead.

The alternative functional architectures could be defined using object-oriented or structural decomposition techniques. The goal is to define the functions (or methods) that the system will perform as well as the information that is needed to perform them.

Entrance Criteria

- Knowledge of applicable technologies exists.
- System users are available.
- Complete external system definitions that will affect design choices (e.g., mandatory/existing reusable off-the-shelf systems subsystems/ components) are available.
- Consistent and complete system requirements specifications exist.

Exit Criteria

- A functional architecture that defines the functions the system must provide, as well as the interactions between the functions, is defined. The functional architecture is consistent and complete.

Inputs

- Configuration Control/Constraints
- Evaluation Results
- External System Definition
- System Definition Interim
- System Definition/Process Requirements

- System Requirements
- System Validation Results
- Technology Base
- User Requirements
- Validation & Verification Results

Outputs

- Alternative Functional Architectures
- Derived Requirements

B.2.1.2.3.1 Partition Requirements Into Functions

The Partition Requirements Into Functions activity (1.2.3.1) takes the system requirements and partitions them into a set of functions that define the allocation-independent system design. These functions are not necessarily mapped neatly to the informal functions defined earlier but are directly derived from the system requirements. The partitioning is based on the behavioral requirements but influenced by the performance requirements.

During the Partition Requirements Into Functions activity (1.2.3.1), the system requirements are partitioned into a set of functions that define the allocation-independent system design. The partitioning of functions is based on the degree of interaction between the behavioral requirements as well as the mapping of performance requirements to the behavioral requirements. Another influencing factor in the partitioning of the system requirements into functions is design constraints. If there are reuse requirements, or commercial off-the-shelf (COTS) systems that must be used, some of the partitioning may be based on these decisions. Finally, there may be political considerations that constrain the way the requirements are partitioned.

Entrance Criteria

- Knowledge of applicable technologies exists.
- System users are available.
- Complete external system definitions that will affect design choices (e.g., mandatory/existing reusable off-the-shelf systems subsystems/ components) are available.

Exit Criteria

- A consistent and complete set of partitions that capture the requirements is defined.

Inputs

- Alternative Functional Architectures < Interim >
- Functional Hierarchy < Interim >

Outputs

- Derived Requirements
- Partitioned Requirements

B.2.1.2.3.2 Define Lower Level Functions

The Define Lower Level Functions activity (1.2.3.2) decomposes the functions defined from the partitioned system requirements into lower level functions. These functions are defined as the result of design decisions that begin to specify how the system will satisfy the system requirements. These design decisions should still be allocation-independent but begin to constrain the allocation decisions.

The Define Lower Level Functions activity (1.2.3.2) decomposes the functions defined from the partitioned system requirements into lower level functions. The lower level functions begin to define methods and techniques for accomplishing the system requirements. The lower level functions are not intended to define an allocated solution but are intended to define a logical architecture that specifies a set of interacting functional entities that satisfy the system requirements. The lower level functions specify constraints on the allocation that further limit the possible solution, but there is still a range of solutions that could be defined to satisfy the functional architecture.

Entrance Criteria

- Knowledge of applicable technologies exists.
- System users are available.
- Complete external system definitions that will affect design choices (e.g., mandatory/existing reusable off the shelf systems subsystems/ components) are available.
- Partitioned requirements are available.

Exit Criteria

- A consistent and complete functional hierarchy is defined.

Inputs

- Alternative Functional Architectures < Interim >
- Partitioned Requirements

Outputs

- Derived Requirements
- Functional Hierarchy
- Functional Hierarchy Interim

B.2.1.2.3.3 Define Partition/Functional Interfaces

The Define Partition/Functional Interfaces activity (1.2.3.3) defines the interfaces between the partitions and functions that compose the functional architecture. In this activity, the information that flows between the functions is defined as well as the behavior of the interfaces.

The Define Partition/Functional Interfaces activity (1.2.3.3) identifies and describes the interfaces between the functions identified in the functional hierarchy. In this activity, each of the functions is assessed to determine the information that is used to produce the output of the function. From this analysis, the information flows between functions and to the environment are identified. The behavior of the interfaces is also specified (e.g., whether the data is buffered, the anticipated frequency of the external data flow, and the dependence of the functions [e.g., a function requests or responds to data]).

Entrance Criteria

- Functional Hierarchy is available.
- Traceability from functions back to system requirements is understood.

Exit Criteria

- A consistent and complete alternative functional architecture is defined.

Inputs

- Functional Hierarchy

Outputs

- Alternative Functional Architectures
- Alternative Functional Architectures Interim

B.2.1.2.4 Synthesize Allocated Architecture

The Synthesize Allocated Architecture activity (1.2.4) uses the system requirements, the alternative functional architectures, and input from the user/customer to produce the alternative allocated architectures.

Figure 55 (Appendix A) contains the decomposition of the Synthesize Allocated Architecture activity (1.2.4). During this activity, the alternative functional architectures are allocated to alternative allocated architectures. The allocated architectures are potentially physical architectures that define the viable alternatives. This may result in the physical layout of the system, or only the logical subsystems, and their interactions. If the solution defines a physical architecture, then the functions are allocated to hardware, software, and people (procedures). Interfaces are defined that communicate the interactions between the parts in the allocated architecture, and technical parameters are defined that drive the performance parameters of the allocated parts. Together, the allocation, interfaces, and technical parameters define the allocated architecture.

In this activity, the Integrate Design activity (1.2.4.4) calls out the integration of the subsystem and component designs that define the system. This design integration allows reevaluation of system designs using more accurate information provided by the lower level designs. This means that design problems can be found much earlier in the system definition process saving both time and money. This activity is also used to communicate across the system to subsystems and components that have a direct bearing on the system under consideration. As an example, the drive shaft must be producible by the drive shaft pressing machine. As a result, the the drive shaft pressing machine must provide input to the Integrate Design activity.

Entrance Criteria

- Knowledge of applicable technologies exists.
- System users are available.
- Complete external system definitions that will affect design choices (e.g., mandatory/existing reusable off-the-shelf systems subsystems/ components) are available.
- Consistent and complete system requirements specifications are available.
- At least one alternative functional architecture is defined.

Exit Criteria

- Consistent and complete alternate architectures that appear to satisfy the system requirements are defined. These architectures define the subsystems, hardware, software, and/or procedures that define the system design. All of the performance requirements allocations are documented in the architecture.

Inputs

- Allocated Requirements & Performance Estimates
- Allocated Technical Parameters
- Alternative Functional Architectures
- Configuration Control/Constraints
- Evaluation Results
- External System Definition
- Sensitivity Analysis Results
- System Assessment
- System Definition Interim
- System Definition/Process Requirements
- System Requirements
- Technical Risks and Problem Assessment
- Technology Base
- Trade-Off Results
- User Requirements
- Validation & Verification Results

Outputs

- Alternative Allocated Architectures
- Derived Requirements

B.2.1.2.4.1 Allocate Functional Architecture to Alternative Solutions

During the Allocate Functional Architecture to Alternative Solutions activity (1.2.4.1) the alternative functional architectures, which consist of the functional decomposition, hierarchy, and interfaces, are allocated to potential system solutions. Recognizing that functions can be accomplished via different means, each alternative solution allocates the functionality to different implementing mechanisms.

The Allocate Functional Architecture to Alternative Solutions activity (1.2.4.1) defines multiple system architectures that seem viable. These architectural allocations define alternate physical

architectures and the logical interactions between the allocated functions. Candidate mechanisms for carrying out the functions can be identified from the technology base (in the case of off-the-shelf systems). The mechanisms are partitioned into assemblies or subassemblies based on the functional architecture and the allocated performance characteristics.

Entrance Criteria

- A complete and consistent functional architecture is available.
- Generalized architectural definitions that can be applied to the system have been identified.
- User requirements are available.

Exit Criteria

- A physical architecture that defines subsystems, components, and/or elements is defined.

Inputs

- Alternative Functional Architectures
- Evaluation Results
- Physical Interfaces
- Physical Parameters

Outputs

- Physical Architecture Allocations

B.2.1.2.4.2 Define Physical Parameters

The Define Physical Parameters activity (1.2.4.2) defines the algorithms and parameters that drive the architecture that is selected. These parameters define acceptable ranges of values that can be used to drive the performance of an assembly or subassembly.

The Define Physical Parameters activity (1.2.4.2) defines specific characteristics of an identified logical or physical entity. Physical entities include hardware, software, or people (procedures). The technical parameters drive the performance of the logical or physical entities. The characteristics identified include performance, size, mass, and timing.

Entrance Criteria

- Consistent and complete architecture allocations are available.
- Performance requirements and constraints are available.
- Algorithms that define performance characteristics have been identified.

Exit Criteria

- Physical parameters that are can be used to drive algorithms that indicate whether the system in question can meet the performance constraints (e.g., mass, heat, speed, etc.) are defined.

Inputs

- Evaluation Results
- Integrated Alternative Solutions

- Physical Architecture Allocations
- Physical Interfaces

Outputs

- Physical Parameters

B.2.1.2.4.3 Define Physical Interfaces

The Define Physical Interfaces activity (1.2.4.3) identifies the actual physical interfaces that are needed to support the logical interfaces defined by the mapping of the formal functions and interfaces to the alternate solutions.

Following identification of the potential parts of the system, the Define Physical Interfaces activity (1.2.4.3) determines the necessary interfaces between the parts of the system. The interfaces may be physical interfaces and could be as simple as placement of components in a subsystem, or as complex as determining the best approach to providing for multiple functional interfaces through physical means, possibly in a distributed fashion. The physical interfaces define the data structures and standards used to communicate the data.

Entrance Criteria

- Physical parameters are available.
- Physical architecture allocations are available.
- System performance requirements are available.

Exit Criteria

- A mapping of logical interfaces to physical interfaces in the physical architecture allocations is defined.

Inputs

- Integrated Alternative Solutions
- Physical Architecture Allocations

Outputs

- Physical Interfaces

B.2.1.2.4.4 Integrate Design

The Integrate Design activity (1.2.4.4) takes the "as built" designs (if they exist) from the subsystems or components that are created in lower level or peer system definitions and uses the results to update the system architecture.

The Integrate Design activity (1.2.4.4) integrates the actual system definition from all lower level designs within its purview. If a subsystem has allocated parameters to lower level components, this activity takes the actual design information and incorporates it into the subsystem design.

Entrance Criteria

- At least one available external system definition that includes "as defined" physical architecture allocation, including physical parameters, interfaces, physical architecture allocations, physical parameters, and physical interfaces for subsystems or components, is available.

Exit Criteria

- Alternate solutions with "as defined" subsystems and/or components integrated and any derived requirements for those alternate solutions are defined.

Inputs

- Alternative Allocated Architectures
- External System Definition
- Physical Architecture Allocations
- Physical Interfaces
- Physical Parameters

Outputs

- Derived Requirements
- Integrated Alternative Solutions

B.2.1.2.4.5 Define Physical Architecture

The Define Physical Architecture activity (1.2.4.5) takes the architecture defined by the allocation, interfaces, and parameters and addresses design considerations that result from the chosen solution.

During the Define Physical Architecture activity (1.2.4.5), each solution is defined to a level of actual physical and/or concrete characteristics. This is basically a definition of how the functional architecture will be performed. In defining a physical architecture, decisions must be made as to how functions will be performed, including preliminary determination as to a mechanical, electrical, software, or human solution. The resultant architecture may introduce other functionality that is created to deal with the consequences of the allocation. This includes the identification of failure mechanisms and other entities identified to support startup, restart, reconfiguration, and degraded operation.

Entrance Criteria

- Physical architecture allocations with associated physical parameters and physical interfaces are available.

Exit Criteria

- Consistent and complete alternate architectures that appear to satisfy the system requirements are defined. These architectures define the subsystems, hardware, software, and/or procedures that define the system design. All of the performance requirements allocations are documented in the architecture.

Inputs

- Evaluation Results
- Integrated Alternative Solutions

Outputs

- Alternative Allocated Architectures
- Derived Requirements

B.2.1.2.5 Evaluate Alternatives

The Evaluate Alternatives activity (1.2.5) assesses the various alternative allocated architectures, performs sensitivity analysis, allocates technical parameters, identifies the technical risks and

problems, evaluates the alternately allocated architectures against each other, and selects the preferred system solution.

Figure 56 (Appendix A) contains the decomposition of the Evaluate Alternatives activity (1.2.5). This activity assesses the system, performs sensitivity analysis, allocates technical parameters, identifies and assesses technical risks and problems, identifies and performs trade-offs, and selects the system solution.

The assessment of the system is separate from the sensitivity analysis and trade-offs. The assessment activity assesses its various performance characteristics. During the sensitivity analysis, an attempt is made to optimize the system performance characteristics for a particular system design. In trade-off analysis, alternate optimized system designs are evaluated against each other.

Entrance Criteria

- Alternative Allocated Architectures are available.
- Approved system definition is available.
- User requirements are available.
- Characteristics of available technologies are identified.
- Specific engineering domain or physics algorithms that define performance or behavior of functions or sets of functions are available.

Exit Criteria

- The preferred system solution is selected.
- Allocated requirements and performance estimates are complete.
- Evaluation results that define the performance characteristics of the alternate architectures evaluated are documented.
- Technical risks associated with the various solutions and an assessment of the problems associated with the risks are identified.

Inputs

- Alternative Allocated Architectures
- Configuration Control/Constraints
- Evaluation Documentation < Baselined >
- External System Definition
- Performance Requirements
- System Definition < Interim >
- System Definition/Process Requirements
- Technology Base

- User Requirements
- Validation & Verification Results

Outputs

- Allocated Requirements & Performance Estimates
- Evaluation Results
- System Solution < Preferred >

B.2.1.2.5.1 Assess System

The Assess System activity (1.2.5.1) defines models to be used to evaluate the system as a whole. During this activity, each of the performance analysis disciplines (the "ilities") build models to assess the system against the performance characteristics that affect that discipline.

The Assess System activity (1.2.5.1) assesses the entire system that is being developed. Under normal circumstances, it is impractical to continuously analyze the full scope of the system, so analysis is performed on subsystems or components only. In this activity, a total picture must be taken in order to ensure the analysis of all elements of the system, many of which will not fit neatly into a subsystem or component element. In many cases, it is this system assessment that identifies system weaknesses or missed interfaces. It is preferable that any system assessment be based on a well defined set of technical performance measures (TPMs).

In this activity, the specialty evaluation disciplines are exercised to create models used to evaluate the performance of the system in that domain. This may include air vehicle lift characteristics, reliability, and reusability. Many times the evaluated solution defines parameters that are used along with parameters external to the system in algorithms defined for the domain in question. The goal of the activity is to capture and understand all of the issues that are important to each of the disciplines that are critical to the system.

Entrance Criteria

- Alternative architectures are available.
- An approved system definition is available.
- User requirements are available.
- Characteristics of available technologies are identified.
- Specific engineering domain or physics algorithms that define performance or behavior of functions or sets of functions are available.

Exit Criteria

- A system assessment of evaluated architectures is complete. This assessment includes varied orthogonal evaluation results along particular performance vectors.
- Models that were used to assess the varied performance vectors are documented.

Inputs

- External System Definition
- System Solution < Interim >

Outputs

- System Assessment

B.2.1.2.5.2 Identify and Assess Technical Risks and Problems

The Identify and Assess Technical Risks and Problems activity (1.2.5.2) takes the system assessment and identifies the system's problems and technical risks. The evaluation results are used to identify the technical risks associated with the system solution.

The Identify and Assess Technical Risks and Problems activity (1.2.5.2) determines technical risks and problems and evaluates those risks to determine their potential impact to the system. The identified risks and problems can then be mitigated. As a result of this activity, key design issues are identified that must be resolved to produce a successful system.

Entrance Criteria

- User requirements are available.
- Customer requirements are available.
- System assessments and the associated alternative architectures are available.

Exit Criteria

- Technical risks associated with the various solutions and an assessment of the problems associated with the risks are identified.

Inputs

- Allocated Technical Parameters
- System Assessment
- Trade-Off Results

Outputs

- Technical Risks and Problem Assessment

B.2.1.2.5.3 Perform Sensitivity Analysis

The Perform Sensitivity Analysis activity (1.2.5.3) evaluates the system with respect to variations in system parameters and external environment interactions. This evaluation may affect the assessment of technical risks and problems.

The Perform Sensitivity Analysis activity (1.2.5.3) analyzes the system's sensitivities to varying environments and design parameters, i.e., the impact on system capability due to a change of design functionalities or variabilities in system concepts.

Entrance Criteria

- System assessments along with alternative architectures and associated risks and problems are available.
- Models that were used to assess the various performance vectors are available.

Exit Criteria

- A set of sensitivity analysis results that define the critical ranges of identified physical parameters is complete and documented.

Inputs

- Allocated Technical Parameters

- System Assessment
- Technical Risks and Problem Assessment
- Trade-Off Results

Outputs

- Sensitivity Analysis Results

B.2.1.2.5.4 Allocate Performance to Technical Parameters

The Allocate Performance to Technical Parameters activity (1.2.5.4) defines the values of the technical parameters for the subsystems/components/elements that define the actual selected allocated architecture. The analysis that defines the performance of the system against the critical characteristics within the evaluating discipline occurs in this activity.

During the Allocate Performance to Technical Parameters activity (1.2.5.4), technical parameters, including all physical characteristics and performance/operating requirements, are defined for the allocated architecture. This allocation of characteristics and capability is determined through analysis techniques that attempt to partition system characteristics in a logical and optimal fashion. The result is a large number of orthogonal evaluation results that all summarize the performance and behavior of the system in question. Any external system definitions or implementation results can be used in this evaluation to get a more accurate evaluation.

The evaluation results are correlated against relevant technical performance measures for the system. The goal is to optimize the alternative architecture against the technical performance measures to give the most desirable performance characteristics attainable by the alternative allocated architecture under consideration.

Entrance Criteria

- User requirements are available.
- Customer requirements are available.
- System performance requirements are available.
- Models that were used to assess the various performance vectors are available.
- Sensitivity analysis results are available.

Exit Criteria

- Allocated technical parameters that define the "optimized" set of parameters for each of the alternative system architectures are defined.

Inputs

- Performance Requirements
- Sensitivity Analysis Results
- System Assessment
- Trade-off Results

Outputs

- Allocated Technical Parameters

B.2.1.2.5.5 Identify and Perform Trade-Offs

The Identify and Perform Trade-Offs activity (1.2.5.5) is extremely broad in nature. It performs an analysis to determine the best of multiple potential solutions.

The Identify and Perform Trade-Offs activity (1.2.5.5) identifies the viable system alternatives and completes a trade-off analysis of the systems in question. Each system is assessed against the technical performance measures, the level of risk, and the problems associated with the system. The trade studies done assess the effectiveness of a particular solution against the system measures of effectiveness and communicate information back to the Allocate Performance to Technical Parameters activity (1.2.5.4) to optimize the solution along the assessment parameters.

Entrance Criteria

- User requirements are available.
- Customer requirements are available.
- System assessments along with alternative architectures and associated risks and problems are available.
- Models that were used to assess the various performance vectors are available.
- Sensitivity analysis results are available.
- Allocated technical parameters are available.

Exit Criteria

- Trade-off results that define shortcomings and benefits of the alternate architectures are complete and documented.

Inputs

- Allocated Technical Parameters
- Sensitivity Analysis Results
- Technical Risks and Problem Assessment

Outputs

- Trade-Off Results

B.2.1.2.5.6 Select System Solution

During the Select System Solution activity (1.2.5.6), the system trade-off results are evaluated to select the preferred solution.

The Select System Solution activity (1.2.5.6) is an analysis to determine the optimal solution among many alternatives. This may be an analysis to select an individual alternative, or it may be an analysis to consider the melding of various alternatives into a new integrated alternative.

Entrance Criteria

- User requirements are available.
- Customer requirements are available.

- System assessments along with alternative architectures and associated risks and problems are available.
- Trade-off results are available.

Exit Criteria

- The preferred system solution is defined.

Inputs

- Sensitivity Analysis Results
- System Assessment
- Trade-Off Results

Outputs

- Outputs System Solution < Interim >
- System Solution < Preferred >

B.2.1.2.6 Validate and Verify Solution

The Validate and Verify Solution activity (1.2.6) defines the test procedures for verifying and/or validating a work product. This activity also performs these test procedures to assess the system's measure of effectiveness against the user/customers stated needs and against the previously generated constraints defined in activities prior to the activity who's work products are being evaluated.

Figure 57 (Appendix A) contains the decomposition of the Validate and Verify Solution activity (1.2.6). This activity defines the test procedures that are used to verify and/or validate a work product. This activity also implements these test procedures to actually verify and/or validate the work products. Any deviation from the expected results will be documented.

Verification of the work products occurs when the work product is assessed against the inputs that were used to produce the work product. It is verified that the intent of the input work products is addressed. Test procedures are created to ensure this assessment is complete.

This activity also implements the analysis of the conformance of the work products to the established standards and procedures to assess the quality of the work products.

Entrance Criteria

- The evaluated work products to be validated and/or verified are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution.
- The constraining work products or user/customers that provide the basis for the validation and/or verification are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution, user requirements, and customer requirements.

Exit Criteria

- Validation and/or verification results that assess the evaluated work products conformance to the constraining work products are complete and documented.

Inputs

- Alternative Allocated Architectures
- Alternative Functional Architectures
- Configuration Control/Constraints
- Informal Functional Hierarchy
- System Context
- System Definition < Interim >
- System Requirements
- System Solution < Preferred >
- User Requirements

Outputs

- V&V Test Procedures
- Validation & Verification Results

B.2.1.2.6.1 Define Validation and Verification Test Procedures

The Define Validation and Verification Test Procedures activity (1.2.6.1) utilizes the work product definitions to define test procedures that verify and/or validate the work products.

The Define Validation and Verification Test Procedures activity (1.2.6.1) develops a clear definition of how the validation and verification is accomplished. In many cases, these procedures identify some form of V&V compliance matrix or are integrated into a system test plan. In any case, procedures are defined so that an analysis is performed with an established set of criteria. These tests must ensure that work products conform to established quality measures. These procedures also document any quality assurance checks and balances that must be assessed.

Entrance Criteria

- The evaluated work products to be validated and/or verified are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution.
- The constraining work products or user/customers that provide the basis for the validation and/or verification are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution, user requirements, and customer requirements.

Exit Criteria

- Validation and Verification Test procedures that define the work product verification and/or validation are defined.

Inputs

- System Validation Results
- System Verification Results

Outputs

- V&V Test Procedures

B.2.1.2.6.2 Validate System

The Validate System activity (1.2.6.2) assesses the work product to determine whether the customer/users' needs are being addressed. Test procedures are created to ensure this assessment is complete.

The Validate System activity (1.2.6.2) answers the question, "Are we developing the right system?" This activity ensures that the requirements for the system actually address the needs as stated by system stakeholders. This activity implements the validation test procedures and documents any variation between the expected results and the actual results. A root cause analysis is performed or initiated.

Entrance Criteria

- The work products to be validated are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution.
- The constraining work products or user/customers that provide the basis for the validation are available. These constraints may include the use of the product or the operational scenarios.

Exit Criteria

- Validation results that assess the work product's conformance to the constraining work product are complete and documented.

Inputs

- V&V Test Procedures

Outputs

- System Validation Results

B.2.1.2.6.3 Verify System

The Verify System activity (1.2.6.3) verifies the work product against the previous activities work products. Any activities output work products can be verified against the input work products used to create it.

The Verify System activity (1.2.6.3) ensures that the performance characteristics and constraints defined in all activities leading to the system definition are satisfied by the system. This activity implements the verification test procedures and documents any variation between the expected results and the actual results. A root cause analysis is performed or initiated. This activity also verifies that the defined systems engineering process is adhered to during the definition of the system.

Entrance Criteria

- The evaluated work products to be verified are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution.
- The constraining work products or user/customers that provide the basis for the verification are available. These may include problems and needs, identified environments, informal functional hierarchy, system requirements, alternative architectures, and/or preferred system solution, user requirements, and customer requirements.

Exit Criteria

- Verification results that assess the work products conformance to the constraining work product are complete and documented.

Inputs

- V&V Test Procedures

Outputs

- System Verification Results

B.2.1.2.7 Control Technical Baseline

The Control Technical Baseline activity (1.2.7) maintains and versions the technical decision data that gives the rationale for technical decisions that have been made. This activity also maintains the technical work products configuration control.

Figure 58 (Appendix A) contains the decomposition of the Control Technical Baseline activity (1.2.7). This activity maintains and versions the technical decision data that provides rationale for technical decisions that have been made. This activity also maintains the technical work products configuration control. This activity identifies the configuration items that constitute each historical baseline.

Entrance Criteria

- Work products to be baselined exist.
- The technical system definition process has been initiated.

Exit Criteria

- Configuration control constraints are defined, and/or the current technical baseline and the rational for technical decisions are captured.
- The increment status is available.

Inputs

- Allocated Requirements & Performance Estimates
- Alternative Allocated Architectures
- Evaluation Results
- Informal Functional Hierarchy
- System Definition < Interim Updated >
- System Solution < Preferred >
- V&V Test Procedures
- Validation & Verification Results

Outputs

- Configuration Control/Constraints
- Evaluation Documentation < Baselined >
- Increment Status

- System Definition < Interim >
- Technical Risk

B.2.1.2.7.1 Control Technical Decision Data

The Control Technical Decision Data activity (1.2.7.1) maintains the critical decision data that determined the selected system definition.

The Control Technical Decision Data activity (1.2.7.1) controls critical information pertaining to system definition decisions. This data is periodically baselined. This decision data serves as a library of key system development events and the key decisions made that have resulted in the current system. The data maintained by this activity is usually limited to information that may be needed by other system development activities, management, or reviewers. Data that is not critical to the program is usually maintained in a separate, inactive library.

Entrance Criteria
- Decision data that provides rationale for decisions made exists and is not baselined. This information can come with any of the work products.

Exit Criteria
- Decision data is captured and made available.

Inputs
- System Definition < Interim >

Outputs
- Decision Database
- Evaluation Documentation < Baselined >

B.2.1.2.7.2 Control System Configuration

The Control System Configuration activity (1.2.7.2) maintains configuration control of all of the baselined system definitions. These system configurations may be intermediate ones that do not go to management.

The Control System Configuration activity (1.2.7.2) controls the baselined system definitions. As changes to the system are made at various levels, it is necessary to control a single baseline. This baseline contains information for all areas of system development. The baseline also documents the changes that distinguish the baseline from previous baselines. The baselining of information usually follows a well-defined configuration control process. In this process, proposed changes are reviewed and evaluated, the proposed updates are made, and the results of the baseline documented. The rigor at which this process is performed depends on the maturity of the system description. Early in a program, light control may be maintained to ensure that changes can be incorporated easily, but, as the system matures, the control of the configuration tightens in order to prevent changes that may have a ripple effect through the system. The level of control is also dependent on the level of system decomposition and the determination of the impact of any given change.

Entrance Criteria
- Inputs to the activity exist and are not controlled.
- The technical system definition process has been initiated.

Exit Criteria

- Configuration control constraints are defined, and/or the current technical baseline is captured and available.
- The increment status is available.

Inputs

- Evaluation Documentation < Baselined >

Outputs

- System Definition < Interim >

APPENDIX C. GSEP WORK PRODUCT DESCRIPTIONS

This appendix contains descriptions of GSEP work products and, where appropriate, lists subproducts that help define the product. All work products are listed in alphabetical order.

ALLOCATED REQUIREMENTS & PERFORMANCE ESTIMATES

Allocated Requirements & Performance Estimates contain the results of traceability from the current system design to the requirements. The requirements are allocated across the system architecture. The performance estimates define a breakdown of the performance requirements that are allocated to architectural components in order to carry out the performance requirements. Performance is broadly defined to include reliability, temporal behavior, and physical characteristics (e.g., size, temperature, etc.). The performance estimates are generated from system models that focus on aspects of system performance that is the concern of a particular speciality discipline. These system models are used to evaluate the measure of effectiveness of the particular system design under evaluation.

ALLOCATED TECHNICAL PARAMETERS

Allocated Technical Parameters define the system performance. They include parameters such as power, heat, and timing. A particular solution defines certain parameter boundaries that imply boundaries of other technical parameters. These technical parameters include the acceptable boundary conditions of the performance parameter and the design component(s) that meet the performance objectives defined by the parameters.

ALTERNATIVE ALLOCATED ARCHITECTURES

Alternative Allocated Architectures define mappings from the functional architecture to subsystems, components, people, hardware, and software. They define where the functions are accomplished and how the interfaces between the people, hardware, and software support the interfaces between the formal system functions. This work product includes the physical layout and potential failure mechanisms. Note that this architecture may contain an allocation to subsystems only. In essence, there may be no real physical aspects to the allocated architecture.

Subproducts that help define the product:

Physical Architecture Allocations

Physical Interfaces

Physical Parameters

ALTERNATIVE FUNCTIONAL ARCHITECTURES

The Alternative Functional Architectures define the hierarchy of functions that satisfy the system requirements. Each function documents the inputs to, outputs from, and internal behavior of the function. There are often performance requirements that are allocated to constrain the behavior of the function. Functional interfaces define the information that flows between functions. These interfaces can be electrical, mechanical, or logical. Interfaces define the interactions of the functions with each other as well as the external environment. It should be noted that the functions can be identified using object-oriented or structural decomposition techniques. If object-oriented techniques are used inappropriately, it may be difficult to reuse existing subsystems or components. If structural decomposition is used, the system may be constrained unnecessarily by past systems.

Subproducts that help define the product:

Partitioned Requirements

Functional Hierarchy

APPROACH DEFINITION

The Approach Definition identifies the key stakeholders and their expectations, objectives for the system and this increment, alternative ways that the objectives could be met and the stakeholders satisfied, and the constraints under which the objectives must be met. Objectives must be results oriented, trackable, clear and concise, controllable, realistic, and appropriate. Constraints include external influences, available resources, existing technology, and applicable standards.

ASSET DEVELOPMENT GOALS

The Asset Development Goals define the organization's objectives for reusable assets that can be developed during the creation of a system. These assets may be used during the development of other systems that the organization creates. These assets might include facilities, people, components, subsystems, and other assets.

BEHAVIORAL/PERFORMANCE REQUIREMENTS MAPPING

The Behavioral Requirements Mapping defines the set of behavioral requirements that each performance requirement must constrain. The relationship between these requirements is complex and there can be many different performance requirements that are levied against a behavioral requirement.

BEHAVIORAL REQUIREMENTS

Behavioral Requirements define what the system is supposed to do apart from the qualitative specifications of how well and within what constraints the behavior must be accomplished. The behavioral requirements define the interactions of the system with its environment. These requirements also specify what must be done to produce the system outputs. The behavioral requirements make up part of the system requirements.

CONFIGURATION CONTROL/CONSTRAINTS

Configuration Control/Constraints define the procedures and latitude that the system engineer is allowed in modification to and creation of system description documentation. For work products that are indifferent states, the constraints may vary.

CUSTOMER REQUIREMENTS

Customer Requirements document the customer's expectations for the system being built. The customer specifies the operational characteristics of the system and the performance characteristics. These requirements may include how the system is to be built in addition to what the system must do. The customer is responsible for purchasing the system and must agree to the contractual agreements under which the system will be purchased.

DECISION DATABASE

The Decision Database captures the decisions and rationale for decisions that are made in defining the various work products that make up the system definition.

DERIVED REQUIREMENTS

Derived Requirements document requirements that result from an analysis of user or customer requirements or imposed constraints on the system design. Derived requirements never result from design decisions that are made by the engineering team (though the customer may stipulate design decisions as requirements).

DEVELOPMENT ALTERNATIVE

The Development Alternative documents the development approach for the increment.

ESTIMATE OF THE SITUATION

The Estimate of the Situation defines the mission of the system and its development, the relationships of the organizations involved in the system development, and relationships with stakeholders. It documents objectives, assumptions, and constraints on the development of the system. The objectives can be political, technical, organizational, and/or economic. Assumptions include stakeholder expectations, how interactions are to be handled with other organizational units, and how the system will be staffed.

EVALUATION DOCUMENTATION

The Evaluation Documentation contains the evaluation results, allocated requirements, and performance estimates. The rationale for selection of the system design that will be implemented is also included.

Subproducts that help define the product:

Evaluation Results

Allocated Requirements and Performance Estimates

Decision Database (partial)

EVALUATION RESULTS

The Evaluation Results include the models, test results including sensitivity analysis, and assessed risks identified in the analysis of each of the system alternative designs considered. These evaluation results define estimates of system performance.

Subproducts that help define the product:

Sensitivity Analysis Results

System Assessment

Technical Risks and Problem Assessment

Trade-Off Results

EXTERNAL SYSTEM DEFINITION

The External System Definition defines the system definition of a system external to the system being built. The external system definition comes from higher level systems, or from peer subsystems or components.

EXTERNAL SYSTEM PLAN

The External System Plan defines the constraints that must be used in completing this system's plan, or defines planning issues that must be factored into planning. These could come from a higher level system (for which this system is a subsystem or component). They could also come across from other subsystems that may define planning constraints (e.g., the actual component must be defined before the component test equipment can be defined).

FUNCTIONAL HIERARCHY

The Functional Hierarchy defines the decomposition of functions that define the allocation-independent system design. These functions are decomposed as defined in the method used to partition and decompose the functions. It is possible that the functions are

defined and categorized using an object-oriented method. A subfunction can support multiple higher level functions so the functional hierarchy is more like a network than a hierarchy.

IDENTIFIED ENVIRONMENTS

Identified Environments define the operational environment(s) for the system.

IDENTIFIED PROBLEMS AND NEEDS

The Identified Problems and Needs define the customer's and stakeholder's goals for the system from its conception until it is decommissioned. They also define the reasons for the system's existence. The problems and needs define the operational concept that describes how the system is intended to function, the measures of effectiveness of the system, and the critical influencing factors.

IDENTIFIED RISKS

The Identified Risks are the risks that are associated with the system over its life cycle, from conception to decommissioning. The risks are categorized based on the type of risk (e.g., technical risk, organizational risk, process risk). The risks are also prioritized based on risk exposure which is a measure that combines the risk's likelihood of occurrence with its potential consequence.

IDENTIFIED STAKEHOLDERS

The Identified Stakeholders are the key individuals who have a vested interest in the system or increment that is being developed. Stakeholders may include the user, customer, developing organization's senior staff, system managers, system developers, and any other groups to whom the system or increment is significant.

INCREMENT PLAN

The Increment Plan documents the development goals and associated success criteria that support the planning objectives for the current increment of the system definition. It defines the estimated size and scope of the development for the current increment; development cost and schedule for each development activity planned for the increment; resources allocated to each development activity in the increment; methods, tools, and facilities needed to complete the increment's development activities; sequence and dependencies between the increment's development activities; and WBS for the development activities in the current increment.

INCREMENT STATUS

The Increment Status defines the level of progress in the development of a particular increment of the system. It includes the progress in terms of measurables from each of the systems engineering activities. It is likely that the status information is tailored to the tools and methods used to enact the systems engineering process. The status includes quality metrics, process compliance metrics, cost and schedule actuals, and the technical performance measures of the increment.

INCREMENT STATUS REPORTS

The Increment Status Reports include current plan-to-actual cost information, schedule progress, and risk information that tracks to risk referents (a measure of acceptable risk for each individual overall risk). The increment status reports are derived from the increment status but only include information that is relevant to system planning. If some of the increment status information is relevant to the increment replanning but does not affect the system plan, then that information is not contained in the increment status report.

INFORMAL FUNCTIONAL HIERARCHY

The Informal Functional Hierarchy defines the hierarchy of functions that satisfy the user/customer needs. Each function documents the inputs to, outputs from, and internal behavior of the function. There are often performance requirements that are allocated to constrain the behavior of the function. The informal functional hierarchy captures the customer needs without rigorously defining a set of requirements. The functions may not be testable and are used to communicate a loose framework of interacting behaviors. Functional interfaces define the information that flows between functions. These interfaces are logical and define the interactions of the functions with each other as well as the external environment.

INTEGRATED ALTERNATIVE SOLUTIONS

The Integrated Alternative Solutions define an alternate physical architecture that takes into consideration the actual designs of subsystems or components that are developed in separate system development efforts.

MARKET COST

Market Cost defines the identified distributed cost that a product must not exceed when introduced into the marketplace or the cost that is dictated by the customer/user.

MEASUREMENT AND STATUS

The Measurement and Status indicate the technical progress that determines the completeness of the system being developed.

Subproducts that help define the product:

The current version of the Increment Plan

METHODS/ALGORITHMS

The Methods/Algorithms are the available methods and/or algorithms required to develop a system or product. These may include those methods developed during previous activities (from the asset repository) or those available from other internal and external sources.

METRICS AND PROCESS DRIVERS

Process drivers are the key factors that are used to tailor a process. These drivers include external and internal drivers. The internal drivers include the amount of risk in the

development of the system, current staff capabilities and experience, internal standards, and tools and methods. The external drivers include customer drivers (e.g., cost, schedule) and environmental drivers (e.g., regulations, laws, politics).

ORGANIZATION PLAN

The Organization Plan is developed by an organization that is developing the system and defines how a system or project is to be developed. It identifies those requirements established by the organization and the resources that must be made available by the organization for the development of systems or products.

PARTITIONED REQUIREMENTS

The Partitioned Requirements define the top-level functions for the functional hierarchy. These functions may or may not map nicely to the informal functions that are defined in the informal functional hierarchy.

PERFORMANCE REQUIREMENTS

Performance Requirements define the qualitative specifications of how well the system is to carry out its behavior without specifying what the system is supposed to do. The technical performance measures can be derived from the performance requirements. The Performance Requirements define things such as timing behavior, reliability, safety, and security, with which the behavioral requirements must be accomplished. The performance measures also define all desirable system characteristics such as weight, cost, etc. The Performance Requirements make up part of the system requirements.

PHYSICAL ARCHITECTURE ALLOCATIONS

The Physical Architecture Allocations define the allocated system designs. The physical architecture maps the functions to implementing mechanisms. This may define the people/procedures, hardware, and software that perform the system functions. The physical architecture is relative to the level of the system under consideration. As a result, the implementing mechanisms may be subsystems and/or components and not actual tangible elements. The physical architecture also identifies the interfaces between the functions and maps them to the implementing mechanisms. The Physical Architecture Allocations also define the mapping of performance parameters to the functions and their corresponding implementation mechanisms.

PHYSICAL INTERFACES

Physical Interfaces define the interactions between the implementing mechanisms (i.e., subsystems, components, elements, people/procedures, software, and hardware).

PHYSICAL PARAMETERS

The Physical Parameters define the characteristics of a physical entity. These parameters may be selected in such a way that some of the parameters are bounded while others are constant.

This allows sensitivity analysis of the resultant physical architecture to the physical characteristics of the system.

POLITICAL CONSIDERATIONS

Political Considerations define the political environment in which the system or product is advocated, funded, built, and operated.

PRODUCT PLAN

The Product Plan identifies what system is to be developed. It identifies the constraints, established by the organization for a specific system or system product family.

PROFIT

The Profit is the amount of money the organization anticipates or makes through the production of a system or system product family.

REGULATORY/STATUTORY REQUIREMENTS

Regulatory/Statutory Requirements are legal aspects of regulations and laws that have been enacted or that are anticipated for the period of the life cycle of a system or product.

RESOURCES

Resources are all available resources required to develop a system, including people, knowledge, facilities, property, etc.

RISK AVERSION STRATEGY

Risk Aversion Strategy is the strategy that may be used to mitigate or avert the risk.

RISK MANAGEMENT PLAN

The Risk Management Plan documents the identified risks, potential risk aversion strategies, selected risk aversion strategies and the rationale for their selection, and the implementation plan for the selected strategies.

SENSITIVITY ANALYSIS RESULTS

Sensitivity Analysis Results capture the ability of the system to meet its performance requirements in the face of varying environmental or system design characteristics. The sensitivity analysis determines the acceptable variations in the technical parameters.

SOCIAL CONSIDERATIONS

Social Considerations define the social factors relating to a system's acceptability to society and their potential impact on society. Society can be considered on a local, state, national, or international level.

SUBSYSTEM/COMPONENT STATUS

The Subsystem/Component status defines the status of any subsystems and/or components that the management team developing the system must track, status, and integrate into their plan.

SYSTEM

The System is an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective. The System includes any required documentation. Note that the System includes the operational, support, deployment procedures, etc.

SYSTEM ASSESSMENT

A System Assessment defines the assessment of the system against the defined measures of effectiveness that define the purpose of the system.

SYSTEM BASELINE

The System Baseline is the approved system description that is maintained and controlled throughout the system development process.

SYSTEM CONTEXT

The System Context defines the system context identified by the technical subprocess of the GSEP.

Subproducts that help define the product:

Identified Problems and Needs

Identified Stakeholders

SYSTEM DEFINITION

The System Description includes an organized collection of data that provides the audit trail of decisions and rationale that shaped the system definition. This information defines the current state of the system development and its evolution. The System Definition also contains the technical performance measures and the measure of effectiveness of the system.

Subproducts that help define the product:

Allocated Requirements & Performance Estimates

Alternative Architectures

Behavioral/Performance Requirements Mapping

Decision Database

Evaluation Results

Informal Functional Hierarchy

System Requirements

System Solution

User Constraints

Validation & Verification Results

SYSTEM DEFINITION/PROCESS REQUIREMENTS

The System Definition/Process Requirements define the information that comes back from the "as built" system (subsystem, or components). The System Definition includes performance characteristics of the real system and more accurate environmental information that is used to refine the system design. The Process Requirements define other input to be used in improving the implemented systems engineering process.

SYSTEM IMPLEMENTATION STATUS

The System Implementation Status defines information that is used to plan for updates to the system definition. When there are issues in the implementation of the system, there will likely be iteration of the system definition.

SYSTEM PLAN

The System Plan contains lessons learned in planning, the actual technical plan that is to be followed in creating the system (both defining all subsystems and components and implementing them), estimating algorithms, and anticipated process drivers. The System Plan also contains descriptions of key work products, the system process definition, and the WBS.

Subproducts that help define the product:

Estimate of the Situation

Increment Plan

Increment Status

Risk Management Plan

SYSTEM REQUIREMENTS

The System Requirements define the behavioral and performance requirements for the system that, when met, satisfy the system developer's obligations in the production of the system.

Subproducts that help define the product:

Behavioral Requirements

Performance Requirements

Behavioral/Performance Requirements Mapping

Derived Requirements

SYSTEM SOLUTION

A System Solution is the preferred alternative allocated architecture that is selected after the trade-off analysis. This solution is then selected based on a comparison of all alternatives. The System Solution is used to further engineer or implement the system

Subproducts that help define the product:

System Definition

SYSTEM STATUS

The System Status defines the current state of the system definition being produced, including the status of any lower level subsystems and/or components. The System Status includes the cost/schedule actuals and quality metrics for the system. The System Status also includes the process drivers that are used to tailor the process.

SYSTEM VALIDATION RESULTS

System Validation Results define the input work products' effectiveness in meeting the customer/user needs. The results document the compliance and deviation from the customer/user needs as well as the root cause for any deviations.

SYSTEM VERIFICATION RESULTS

System Verification Results define the input work products' effectiveness in meeting the previous activities' defined objectives and constraints. The System Verification Results document the compliance and deviation from the previous activities' defined objectives and/or constraints as well as the root cause for any deviations.

SYSTEMS

Systems are the combination of multiple individual systems that an organization develops.

Subproducts that help define the product:

System

TECHNICAL BASELINE

The Technical Baseline is the most recently assembled (and complete) system definition that has been approved. The Technical Baseline includes the identification of changes that distinguish the baseline from previous baselines.

Subproducts that help define the product:

System Definition

TECHNICAL RISK

The Technical Risk includes all of the identified risks that may affect the measure of effectiveness of a system definition. The measure of effectiveness of the system definition is determined during the evaluation of the alternatives. The Technical Risks are those risks that are associated with the alternate architectures or the selected solution.

TECHNICAL RISKS AND PROBLEM ASSESSMENT

The Technical Risks and Problem Assessment define both the technical risks and the assessment of problems that are identified in the evaluated system architecture.

Subproducts that help define the product:

Technical Risk

TECHNICAL STATUS

The Technical Status is the filtered status of the current increment of the system. The technical status contains that information that is important when tracking to the System Plan.

TECHNOLOGY BASE

The Technology Base consists of the reusable assets that the systems engineering organization can use. These assets include problem domain data, facilities, methods, tools, candidate processes and systems, subsystems, and components (potentially including their definitions).

TIME TO MARKET

Time to Market is the length of time to produce the system.

TOOLS

Tools are any techniques or products that can be used to implement the system or automate methods.

TRADE-OFF RESULTS

The Trade-Off Results capture the relative measure of effectiveness between alternate systems.

USER REQUIREMENTS

The User Requirements document the user's expectations for the system being built. The user specifies the way that the system will be used to carry out its mission and performance characteristics. These requirements cover what the system must do.

USER SPECIFICATION

User Specification defines the user's goals for the system from a user's perspective.

Subproducts that help define the product:

Identified Environments

Identified Problems and Needs

V & V TEST PROCEDURES

V & V Test Procedures are the test procedures that are developed to verify and validate the System Definition work products.

VALIDATION & VERIFICATION RESULTS

The Validation & Verification Results document the results of any form of verification or validation completed on any of the work products produced in the creation of the system.

This page intentionally left blank.

APPENDIX D. GSEP MAPPING TO SYSTEMS ENGINEERING STANDARDS

This appendix contains a mapping of the GSEP to two different systems engineering standards: the SE-CMM (Software Engineering Institute 1994) and MIL-STD-499B (Department of Defense 1994).

D.1 MAPPING THE GSEP TO THE SE-CMM

The SE-CMM is an assessment model that allows an organization to evaluate its current systems engineering capabilities and to identify areas for improvement in their current systems engineering processes. At this writing, the SE-CMM is still in draft form. It is being created by an industrial collaboration that is composed of organizations that are responsible for the development of large systems, including Lockheed, Hughes, Loral, Texas Instruments, the Software Productivity Consortium, and the Software Engineering Institute (SEI). An industry steering committee was established to provide guidance and direction to the SE-CMM development team, and broad-based inputs to the model's development were achieved through a series of industry workshops and reviews. An official version of the SE-CMM will be released as version 1.0 at the end of 1994.

The SE-CMM includes 17 process areas that fall into one of three categories: Project Management (Management), Engineering, and Organization. The base practices that are defined for each process area represent the essential practices needed to meet the goals of the process area. The process capability for a process area is determined by evaluating to what extent the associated base practices are met by the organization.

Earlier versions of the model have been piloted by industry participants, including Texas Instruments, Hughes, and UTC/Sikorsky. Lessons learned from the piloting efforts have been incorporated into the current draft version of the model.

Table 3 maps the GSEP to the SE-CMM process areas and base practices to demonstrate GSEP's compliance with this emerging standard. The GSEP Compliance column contains the following information:

- If the GSEP has a specific activity that satisfies the SE-CMM base practices, the activity number and name are listed.
- If the GSEP supports SE-CMM base practices through process tailoring (see Section 4), it is described either generally as Process Tailoring or specifically as Process Specification, Process Instantiation, or Continuous Process Improvement (CPI).)
- If the GSEP supports the SE-CMM base practices through the concepts of product families (See Section 5) or tool support (See Appendix F), then Product Families and/or Tool Support is indicated.

In some cases, the GSEP/SE-CMM mapping is only partial; in these cases an asterisk (*) in the Base Practice column denotes partial GSEP support for the base practice.

Table 3. Mapping of SE-CMM Base Practices to GSEP Concepts and Activities

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
1.00 Analyze Candidate Solutions	1.01 Establish Evaluation Criteria	1.2.5.4 Allocate Performance to Tech. Parameters
	1.02 Define Analysis Approach	1.2.5.5 Identify and Perform Trade-Offs
	1.03 Identify Additional Alternatives	1.2.5.5 Identify and Perform Trade-Offs
	1.04 Analyze Candidate Solutions	1.2.5.5 Identify and Perform Trade-Offs
	1.05 Select Solution	1.2.5.6 Select System Solution
	1.06 Capture Results	1.2.7 Control Technical Baseline 1.2.5.5 Identify and Perform Trade-Offs
2.00 Develop Functional and Performance Requirements	2.01 Develop Detailed Operational Concept*	1.2.1.2 Assess Problems and Needs 1.2.1.3 Define Environment
	2.02 Identify Key Requirement Issues	1.2.2.1 Determine Behavioral Requirements 1.2.2.2 Determine Performance Requirements 1.2.2.3 Map Behavior to Performance
	2.03 Partition Functions	1.2.3 Define Functional Architecture
	2.04 Derive Requirements	1.2.2 Define Requirements 1.2.4 Synthesize Allocated Architecture
	2.05 Develop Interface Requirements	1.2.3 Define Functional Architecture 1.2.1.3 Define Environment
	2.06 Develop System Data Requirements	1.2.3 Define Functional Architecture 1.2.4.3 Define Physical Interfaces
	2.07 Allocate Requirements	1.2.3.1 Partition Requirements Into Functions
	2.08 Ensure Requirement Verifiability	1.2.2.4 Refine Requirements
	2.09 Capture Requirement Traceability and Status	1.2.2.4 Refine Requirements
	2.10 Capture Results and Rationale	1.2.7 Control Technical Baseline

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
3.00 Develop Physical Architecture	3.01 Derive Physical Requirements	1.2.4.2 Define Physical Parameters
	3.02 Identify Key Design Issues	1.2.5.2 Identify and Assess Technical Risks and Problems
	3.03 Develop Physical Structure	1.2.4 Synthesize Allocated Architecture
	3.04 Develop Physical Interface Requirements	1.2.4.3 Define Physical Interfaces
	3.05 Allocate Physical Requirements	1.2.4.5 Define Physical Architecture
	3.06 Capture Traceability and Status	1.2.2.4 Refine Requirements
	3.07 Capture Results and Rationale	1.2.5 Evaluate Alternatives 1.2.7 Control Technical Baseline
4.00 Integrate Disciplines	4.01 Identify Product Development Disciplines	Process Instantiation
	4.02 Train Interdisciplinary Roles	Process Instantiation
	4.03 Develop Interdisciplinary Understanding	1.2.5.1 Assess System
	4.04 Establish Integrated Development Coordination Methods	Process Instantiation
	4.05 Establish Resolution Methods	Process Instantiation
	4.06 Use Interdisciplinary Methods	Process Instantiation
	4.07 Communicate Results	Process Specification
	4.08 Develop Clear Objectives	1.2.4.1 Allocate Functional Architecture to Alternative Solution 1.1.3.4 Commit to Plan 1.1.2.4 Commit to Strategies 1.1.1.2 Estimate Situation 1.1.1.3 Review Context
5.00 Integrate System	5.01 Define Interfaces	1.2.4.3 Define Physical Interfaces
	5.02 Control Interfaces	1.2.7 Control Technical Baseline 1.2.5 Evaluate Alternatives Process Instantiation
	5.03 Verify Receipt of System Elements	Process Specification
	5.04 Verify System Element Correctness	1.2.6.3 Verify System

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
5.00 Integrate System (cont.)	5.05 Verify System Element Interfaces	Process Specification through feedback from implementation
	5.06 Assemble Aggregates of System Elements	Process Specification/Enactment
	5.07 Checkout Aggregate of System Elements*	1.2.5 Evaluate Alternatives 1.2.4.4 Integrate Design System definition of components and subsystems are integrated; implementation is outside GSEP scope.
6.00 Understand Customer Needs and Expectations	6.01 Elicit Needs	1.2.1 Analyze Needs 1.1.2 Analyze Risk 1.1.1 Understand Context
	6.02 Analyze Needs	1.2.1.2 Assess Problems and Needs 1.2.1.3 Define Environment
	6.03 Develop System Requirements	1.2.2 Define Requirements
	6.04 Obtain Concurrence	1.2.2.3 Map Behavior to Performance
	6.05 Inform Customer	0 Develop Systems (outside the GSEP context) 1.1.5.2 Review Progress
7.00 Verify and Validate System	7.01 Establish Verification and Validation Plans	1.1 Manage Development Effort 1.1.5 Develop System Plan Process Instantiation
	7.02 Define Incremental Verification	1.2.6.1 Define V&V Test Procedures
	7.03 Define System Verification	1.2.6.1 Define V&V Test Procedures
	7.04 Define Validation	1.2.6.1 Define V&V Test Procedures Process Instantiation defines methods.
	7.05 Perform and Capture Verification	1.2.6.2 Validate System 1.2.6.3 Verify System
	7.07 Assess Verification and Validation Success	1.2.6.2 Validate System 1.2.6.3 Verify System

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
8.00 Ensure Quality	8.01 Monitor Conformance to Defined Process*	1.2.6.3 Verify System
	8.02 Measure Quality of Work Product	1.2.6.1 Define V&V Test Procedures 1.1.1.2 Estimate Situation
	8.03 Measure Quality of the Process	1.1.4.1 Monitor and Review
	8.04 Analyze Quality Measures*	1.1.5.2 Review Progress
	8.05 Foster Quality Environment	Process Specification
	8.06 Initiate Quality Improvement Activities	1.1.2.4 Commit to Strategies At the design level as opposed to the Process Specification level
9.00 Manage Configurations	9.01 Establish Configuration Management Methodology	Process Instantiation
	9.02 Identify Configuration Units	1.2.7 Control Technical Baseline 1.1.5.1 Product Change Control
	9.03 Maintain Configuration Data	1.2.7 Control Technical Baseline 1.1.5.1 Product Change Control
	9.04 Control Changes	1.2.7 Control Technical Baseline 1.1.5.1 Product Change Control
	9.05 Communicate Configuration Status	1.2.7.2 Control System Configuration 1.2.7.1 Control Technical Decision Data 1.1.5.1 Product Change Control 1.1.5.2 Review Progress
10.00 Manage Risk	10.01 Develop Risk Management Approach	1.1.2.3 Plan Risk Aversion 1.1.3.2 Review Development Alternative
	10.02 Identify Risks	1.1.2 Analyze Risk
	10.03 Assess Risks	1.1.2.1 Perform Risk Analysis
	10.04 Review Risk Assessment	1.1.2.4 Commit to Strategies
	10.05 Execute Risk Mitigation's	1.1.3.1 Execute Risk Aversion
	10.06 Track Risk Mitigation's	1.1.3 Plan Increment Development 1.1.4 Track Increment Development
	10.07 Assess Risk Mitigation's Effectiveness	1.1.2.3 Plan Risk Aversion 1.1.3.2 Review Development Alternative

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
11.00 Monitor and Control Technical Effort	11.01 Direct Technical Effort*	1.1.4.1 Monitor and Review
	11.02 Track Project Resources	1.1.4 Track Increment Development
	11.03 Track Technical Performance Measures	1.1.4.1 Monitor and Review
	11.04 Review Project Performance	1.1.4.1 Monitor and Review
	11.05 Analyze Project Issues	1.1.4 Track Increment Development
	11.06 Control Technical Effort	1.1.4 Track Increment Development
12.00 Plan Technical Effort	12.01 Identify Critical Resources*	1.1.2.1 Perform Risk Analysis
	12.02 Estimate Project Scope*	1.1.1 Understand Context
	12.03 Estimate Project Costs	1.1.5 Develop System Plan 1.1.3 Plan Increment Development
	12.04 Determine Project Process	1.1.1.1 Define Approach
	12.05 Identify Technical Activities	1.1.5 Develop System Plan 1.1.3 Plan Increment Development
	12.06 Define Project Interface	Process Specification
	12.07 Develop Project Schedules	1.1.5 Develop System Plan 1.1.3 Plan Increment Development
	12.08 Establish Technical Parameters	1.2.1.4 Develop Informal Functionality 1.2.2 Define Requirements 1.2.5.4 Allocate Performance to Technical Parameters 1.2.4.2 Define Physical Parameters
	12.09 Develop Technical Management Plan	1.1.5 Develop System Plan 1.1.3 Plan Increment Development
	12.10 Review Project Plans	1.1.3.4 Commit to Plan
	12.11 Commit to Project's As Plans	1.1.3.4 Commit to Plan
13.00 Define Organization's Systems Engineering Process	13.01 Establish Process Goals	Use GSEP to define the process.
	13.02 Collect Process Assets	1.2.7 Control Technical Baseline (System Definition related)

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
13.00 Define Organization's Systems Engineering Process (cont.)	13.03 Develop Organizational Process	Use GSEP to define the process.
	13.04 Define Tailoring the Guidelines	Use GSEP to define the process.
14.00 Improve Organization's Systems Engineering Processes	14.01 Appraise Process	Use GSEP that creates process to do CPI.
	14.02 Plan Process Improvements	Use GSEP that creates process to do CPI.
	14.03 Change the Standard Process	Use GSEP that creates process to do CPI.
	14.04 Communicate process improvements	Use GSEP that creates process to do CPI.
15.00 Manage Product Line Evolution	15.01 Define Product Evolution	Product Family
	15.02 Identify New Product Technologies	Product Family
	15.03 Adapt Development Processes	Product Family
	15.04 Ensure Critical Component Availability	Product Family
	15.05 Manage Product Technology Insertion	Product Family
16.00 Manage Systems Engineering Support Environment	16.01 Maintain Technical Awareness	Tailoring/ Tools Support
	16.02 Determine Support Requirements	Tailoring/ Tools Support
	16.03 Assess Support Environment	Tailoring/ Tools Support
	16.04 Obtain Systems Engineering Support Environment	Tailoring/ Tools Support
	16.05 Tailor Systems Engineering Support Environment	Tailoring/ Tools Support
	16.06 Insert New Technology	Tailoring/ Tools Support
	16.07 Maintain Environment	Tailoring/ Tools Support
	16.08 Monitor Systems Engineering Support Environment	Tailoring/ Tools Support

Table 3, continued

SE-CMM Process Area	SE-CMM Base Practice	GSEP Compliance
17.00 Manage Systems Engineering Training	17.01 Identify Training Needs	Tailoring/ Tools Support
	17.02 Prepare Training Materials	Tailoring/ Tools Support
	17.03 Train Personnel	Tailoring/ Tools Support
	17.04 Assess Training Effectiveness	Tailoring/ Tools Support
	17.05 Maintain Training Records	Tailoring/ Tools Support
	17.06 Maintain Training Materials	Tailoring/ Tools Support

D.2 MAPPING THE GSEP TO MIL-STD-499B

MIL-STD-499B is a systems engineering standard that "is to assist in defining, performing, managing, and evaluating systems engineering efforts in defense system acquisitions and technology developments. The scope and requirements of systems engineering are defined in terms of what should be done, not how to do it. As a result, the systems engineering activities to manage are defined, not how to manage them" (Department of Defense 1994). This standard is currently in the process of revision and is scheduled to be released as an EIA standard, EIA Interim Standard 632, Systems Engineering. Table 4 shows how the GSEP meets the systems engineering process requirements identified in Section 4 of the standard.

The GSEP Activity column shows the activity or calls out report sections that address 499B requirements.

Table 4. Mapping of MIL-STD-499B to GSEP Concepts and Activities

Requirements Category	499B Requirements	GSEP Activity
Requirements Analysis	Analyze Mission and Environments	1.2.1 Analyze Needs
	Refine Customer Objectives and Requirements	1.2.1 Analyze Needs
	Define/Refine Performance and Design Constraint Requirements	1.2.2 Define Requirements
	Define Initial Performance Requirements	1.2.2 Define Requirements
	Identify Functional Requirements	1.2.2 Define Requirements
	Develop Set of Measures of Effectiveness	1.2.1.2 Assess Problems and Needs
Functional Analysis/Allocation	Define/Refine/Integrate Functional Architecture	1.2.3 Define Functional Architecture
	Decompose Top-Level Functions to Lower-Level Functions	1.2.3 Define Functional Architecture
	Allocate Performance and Other Limiting Requirements to All Functional Levels	1.2.3 Define Functional Architecture
	Define/Refine Functional Interfaces	1.2.3 Define Functional Architecture

Table 4, continued

Requirements Category	499B Requirements	GSEP Activity
Functional Analysis/Allocation (cont.)	Iterate Functional Architecture with Requirements Analysis to Define Mission and Environment Driven Performance.	1.2.3 Define Functional Architecture 1.2.2.2 Determine Performance Requirements
	Iterate Functional Architecture with Synthesis to Define/Refine Feasible Solution Alternatives Which Meet Requirements and to Place Derived Requirements Into the Functional Architecture.	1.2.3 Define Functional Architecture Iterate with 1.2.4 Synthesize Allocated Architecture Iterate with 1.2.5 Evaluate Alternatives
Synthesis	Transform Architectures (Functional to Physical)	1.2.4 Synthesize Allocated Architecture
	Define Alternative System Concepts, Configuration Items & System Elements	1.2.4 Synthesize Allocated Architecture 1.2.5 Evaluate Alternatives
	Define/Refine Physical Interfaces (Internal/External)	1.2.4.3 Define Physical Interfaces
	Define Alternative Product and Process Solutions	1.2.4.5 Define Physical Architecture
	Determine Completeness of Functional and Performance Requirements and Identify Derived Requirements	1.2.2 Define Requirements 1.2.3 Define Functional Architecture 1.2.4 Define Physical Parameters
	Identify Critical Parameters; Analyze Variability	1.2.1.2 Assess Problems and Needs 1.2.2.4 Refine Requirements 1.2.5.1 Assess System
	Develop System Description	1.2.3 Define Functional Architecture 1.2.4 Synthesize Allocated Architecture 1.2.5 Evaluate Alternatives 1.2.7 Control Technical Baseline
	Evaluate Alternatives; Apply Design Simplicity Concepts	1.2.5 Evaluate Alternatives Iterate with 1.2.4 Synthesize Allocated Architecture
	Demonstrate/Verify Design	1.2.6 Validate and Verify Solution

Table 4, continued

Requirements Category	499B Requirements	GSEP Activity
Systems Analysis & Control (Balance)	Select Preferred Alternative	1.2.5.6 Select System Solution
	Trade-Off Studies	1.2.5.5 Identify and Perform Trade-Offs
	Effectiveness Analysis	1.2.5.1 Assess System 1.2.5.3 Perform Sensitivity Analysis 1.2.5.4 Allocate Performance to Technical Parameters
	Risk Management	1.1.2 Analyze Risk
	Configuration Management	1.1.5.1 Product Change Control 1.2.7 Control Technical Baseline
	Interface Management	1.1.5.1 Product Change Control 1.2.7 Control Technical Baseline
	Data Management	1.1.5.1 Product Change Control 1.2.7 Control Technical Baseline
	Progress Measurement: TPM Reviews	1.1.4 Track Increment Development 1.2.5.2 Identify and Assess Technical Risks and Problems
	Integrate Technical Disciplines	See Section 3.3.3 in this report for how concurrent engineering is supported by the GSEP.
	Functional & Performance Validity, Consistency, Attainability, Etc.	1.2.6 Validate and Verify Solution 1.2.5 Evaluate Alternatives
Verification	Evaluate Progress and Effectiveness of Evolving System Solutions	1.1.4.1 Monitor and Review 1.1.5.2 Review Progress 1.2.6.1 Define V&V Test Procedures 1.2.6.2 Validate System 1.2.6.3 Verify System
	Measure Compliance with Requirements	1.2.6.1 Define V&V Test Procedures 1.2.6.3 Verify System
	Demonstrate, Test, and Inspect to Verify: Risk Compliance with Requirements Proof of Concept	1.2.5 Evaluate Alternatives 1.2.6 Validate and Verify Solution

Table 4, continued

Requirements Category	499B Requirements	GSEP Activity
Verification (cont.)	Test and Evaluation	1.2.5 Evaluate Alternatives 1.2.6 Validate and Verify Solution
	Product Assurance	Partial in: 1.2.6.2 Validate System 1.2.6.3 Verify System Partial through: Process Specification (see GSEP tailoring)

This page intentionally left blank.

APPENDIX E. ENGINEERING THE GSEP

E.1 OVERVIEW

Systems engineering, as defined in Section 2, is a comprehensive, iterative, problem-solving process that transforms needs into an optimized system solution. If a system can consist of people, products, and processes, then the systems engineering process should be considered a system, and the process can be defined and refined using the systems engineering process itself.

This section documents the steps taken to develop the GSEP, from its conception through its final form, as it is presented in Section 3 and Appendixes A through C of this technical report.

This appendix is composed of the following sections:

- Section E.2 discusses the technical foundations of the GSEP.
- Section E.3 discusses the key activities used to define the GSEP.
- Section E.4 provides an overview of the GSEP process modeling approach and some of the rationale for its selection.

E.2 TECHNICAL FOUNDATIONS OF THE GSEP

E.2.1 Selecting a Process Development Approach

In selecting a process development approach, the GSEP development team decided that GSEP was a system and that a systems engineering approach should be used to develop that system. By doing this, the team felt that the fundamental systems engineering concepts, as identified in Section 2, would be addressed in a systematic way. At the start, there were two identified approaches to accomplishing this goal. The team could select a systems engineering process that was already identified in a well-known reference, or the team could develop a new process that addressed the same issues as other established processes. The team decided to start with existing standards, but refine those standards through the creation of a generic process model.

In *Successful Systems Engineering for Engineers and Managers*, Reilly (1993) discusses the six fundamental systems engineering process paradigms: rapid development, spiral, early prototype, staircase with feedback, staircase (waterfall), and build-test-fix. Of these process paradigms, the GSEP team selected the early prototype process for the creation of GSEP. This decision was reached as a result of the moderate risk involved in generating the process along with the short schedule constraints. This process begins with a prototype of the system, in this case the GSEP prototype, then moves into a rigorous requirements-through-design process, and finishes with test and acceptance.

The process used by the team is unique in that the prototype served as the process that would be used in the follow-on steps. In order to develop a prototype, the team reviewed multiple systems engineering references, selected a systems engineering standard as a basis for developing the GSEP, and continuously coordinated the process development work with other ongoing systems engineering activities.

E.2.2 Review of Systems Engineering References

The GSEP team reviewed a number of influential references on the subject of systems engineering and management, as well as foundations of theory that relate to systems and systems architecting. Key sources included Blanchard and Fabrycky (1981, 1990), James Lacy (1992), and MIL-STD-499B (Department of Defense 1994).

E.2.3 Selecting a Systems Engineering Standard

Much debate surrounded the selection of a systems engineering standard for the process development activity. Of concern to the team was the viability and acceptance of a standard that contained the principles of systems engineering that were important for any generic systems engineering approach. Two standards, both in draft form, met the principle criteria. Those standards were MIL-STD-499B and the industry standard IEEE P1220.

At the time the decision was made to select a standard, 499B had been submitted to the Office of the secretary of Defense for final review and approval, while P1220 had just been released for review. Rumor had it that 499B may be shelved and replaced by a "commercial" standard, EIA Interim Standard 632 (Electronics Industry Association 1994), as is the current trend with DoD standards. However, because the Consortium's Technical Advisory Group (TAG) was strongly in favor of 499B, and the fact that the standard relayed the teams' principle concepts, it was chosen as the baseline standard.

P1220 was not selected because it would have been too high risk to base the GSEP on a standard that was still in the very early stages of its review cycle. Thus, it was felt that mapping a generic systems engineering process to P1220 would be difficult. This is not to say that P1220 did not have any influence on the GSEP. The description of systems engineering as a decision-making process contained a key principle, continuous verification and validation in concert with systems analysis, that is one of the fundamental concepts of GSEP.

The GSEP is 499B-compliant and could be used to derive a set of integrated GSEP instantiations that map to the P1220 standard. Neither of these standards gives a generic systems engineering process view, nor concretely identifies the interactions between the various levels of system decomposition.

Standard 499B presents a list of requirements for the process as well as the concerns that must be addressed in tailoring the process. It also calls out deliverable documents. The GSEP defines the activities to be accomplished and the work products that define how the activities interact. The GSEP does not emphasize the creation of external deliverables more than the creation of internal work products. Also, 499B mixes management and evaluation disciplines in the Systems Analysis and Control activity, and the GSEP defines them as separate activity sets.

The P1220 standard, on the other hand, called out the technical activities in great detail in the technical process but only gave characteristics of management concerns to be addressed rather than calling out the management process and how it interacts with the technical process.

E.2.4 Coordinating With Ongoing Systems Engineering Activities

While the Consortium has been developing the GSEP, it has also been involved in a number of other systems engineering activities and has continually kept abreast of systems engineering development in the government, industry, and academia. The Consortium has been actively involved in various committees established by NCOSE, participating in the development of the SE-CMM.

In addition to the GSEP development effort, the Consortium has undertaken several systems engineering related activities and studies and has been migrating many of its technologies into the systems arena. There has been an assessment of systems engineering related tools (see Appendix F) as well as a look at the trends in systems engineering (see Appendix G) in order to get a feel for where industry thinks systems engineering will be in the future.

E.3 DEFINING THE GSEP

After baselining the prototype process, the team used the prototype GSEP to further develop the Generic Systems Engineering Process. The GSEP prototype is a combined management and technical systems engineering process that is envisioned to exist at any level of system definition. The GSEP was instantiated for the overall systems engineering process definition effort. The "system" in this case was the GSEP.

At the same time the GSEP was being matured, the team needed to define how the GSEP could be used on a real project. The tailoring portion of the report (see Section 4) was concurrently written to address this need. The tailoring addressed the various contexts that the GSEP activities must support and as a result affected the GSEP definition. At the same time, the maturation of the GSEP affected the way that the process could be tailored.

Finally, the GSEP was used to define a high level process for engineering a family of products (see Section 5). This tailoring was used to validate the process as well as introduce highly leveraged reuse in a systems engineering context.

E.3.1 Managing the GSEP Development Effort

In order to fully understand the complexity of the process definition effort, or the development of any system, the process must begin with management activities. The goal of this management effort is to determine project objectives, alternatives, constraints, and risks and transform that context into a manageable plan of action that would be carried out by the technical subprocess. This plan would include tailoring requirements for instantiating the GSEP to the next layer of system decomposition; that is, fully elaborating the appropriate GSEP activities as required for developing the system.

The management activities included in the GSEP were adopted from the Evolutionary Spiral Process (ESP) (Software Productivity Consortium 1994c). ESP provides a framework for integrating key management and development activities necessary for successfully producing a product. The generic ESP model is an evolutionary management model with a strong emphasis on risk management and mechanisms for systematically evolving plans to reflect changing circumstances. In the GSEP, the team structured the activities such that the spiral nature of the process is supported but not enforced. The team followed the GSEP management activities in order to manage development of the GSEP itself.

E.3.1.1 Understand Context

At the beginning of the project, the team identified and answered basic questions about project objectives, stakeholders, alternatives, and constraints. Several objectives were identified for the

project derived from Consortium planning documents, along with budget, schedule, and technical constraints. Stakeholders, or those with a vested interest in the project objectives, ranged from members of the development team to representatives from Consortium-affiliated companies and the systems engineering community at large. Alternative strategies for achieving the objectives were considered.

E.3.1.2 Analyze Risk

Once the team understood the project context, they identified and analyzed risks and determined and agreed to mitigation strategies. For example, objectives were scoped according to budget and schedule constraints, and conflicting expectations between stakeholders were identified and resolved. The result of this activity was consensus from the appropriate stakeholders on project boundaries and intended results, as well as process and technical strategies for achieving the desired results.

E.3.1.3 Increment Plan Development

Four increments of the plan were completed for this report. Each of these increments was reviewed by internal reviewers, and the third increment was reviewed by external reviewers as well. Increment development planning for each of the system increments proceeded in the following way:

- Increment 1 outlined most of the sections and included the GSEP, IDEF0 diagrams, work product descriptions, and (partial) activity descriptions.
- Increment 2 consisted of the complete report, but each of the outputs was published from the respective tools that were used for their creation.
- Increment 3 consisted of the report integrated into one publishable format.
- Increment 4 consisted of the final version of the report that had been through technical editing and clean proofing.

E.3.1.4 Track Increment Development

Each increment was tracked to the Increment Plan although some adjustments were made during the increments, and there were significant personnel shifts.

E.3.1.5 Develop System Plan

Approved strategies, and other decisions made while understanding the project context and analyzing risks, were all used as key inputs into the planning activity. The GSEP team developed a draft outline of the technical report early in the project in order to effectively communicate intended content, scope, and boundaries. The team developed and approved a detailed schedule of technical activities for engineering the GSEP and completing the technical report. The management and development activities captured on the detailed schedule corresponded to the GSEP activities since the team "engineered the process by following the process." The activities were described on the schedule in terms of responsible team member(s), start and end dates, and durations. The activities were described in the prototype GSEP in terms of how they should be performed.

The final project plan consisted of:

- A document that summarized the project context, risks, and risk aversion strategies
- Success criteria for tracking how well the project was meeting its objectives
- The draft technical report outline
- The draft project process definition
- A detailed schedule

These planning documents evolved as the project proceeded.

E.3.2 Technical Definition of the GSEP

In order to fully understand the complexity of the systems engineering process, the full context and interactions within the process must be understood. The goal of the GSEP development effort was to determine process objectives, alternatives, constraints, and risks and transform that context into a generic process that allows the creation of complex systems. The process for defining the GSEP ensured that the GSEP was well engineered. To ensure these goals were met, the team used the prototype model that was based on the earlier referenced materials (see Section E.2.2) to create the GSEP.

The highest risk technical activities were the definition of the problems and needs, the identification of the requirements, and the verification/validation of the process. The following sections detail the way that these activities where carried out when engineering the GSEP.

E.3.2.1 Analyze Needs

Understanding the technical context includes identifying system stakeholders, assessing problems and needs of the stakeholders, and defining the system environment. From a Consortium perspective, it was not difficult to determine the "business" related stakeholders, however, the identification of stakeholders in the systems engineering process itself was nearly impossible. It was simply concluded that the process must be generic in nature and tailorable so that it could apply to any system definition effort within the spectrum of business activities. Therefore, all people involved in the definition, fabrication, deployment, operations, support, etc., of a system may be stakeholders in the development of a systems engineering process. Of course, this definition is impractical, so the stakeholders must be more narrowly defined in terms of the application if a generic process. It was concluded that the stakeholders must be limited to the target audience of this technical report, including:

- Professionals responsible for producing or defining the systems engineering process, including process improvement group members and systems engineering professionals working on process definition activities
- Members of an integrated product development team to aid them in understanding the principles of systems engineering processes and the systems management and integration of complex systems at different levels of system decomposition
- Consortium professionals as a tool for understanding the concepts of the systems engineering process and as a framework for future systems-engineering-related process and method development

E.3.2.2 Define Requirements

Formal requirements were established for the systems engineering process. Some of the requirements were documented in a workshop summary (Software Productivity Consortium 1993b). The requirements that are identified address the problems that were called out in the workshop summary. Some of the workshop requirements were not testable and, as a result, lower level requirements were derived from them. Requirements were also taken from 499B. The focus was the process requirements as opposed to requirements for methods or techniques selected to instantiate the process. Requirements were also derived from the operational scenarios. Finally, there are requirements that relate to the use of the GSEP in Consortium-affiliated companies. These requirements were derived from input such as TAG meetings, visits to Consortium-affiliated companies, and feedback from systems engineers who were external reviewers.

Unfortunately, the requirements traceability was not completed because of time constraints. To ensure that all of the stated requirements are met and the GSEP is well engineered, this activity must be completed as the GSEP matures.

E.3.2.3 Define Functional Architecture

In the creation of the functions, the team identified the following four reasons that functions were created:

- Exposure of interfaces that help to address current systems engineering process issues
- Exposure of interfaces that relate to the communication of important domain concepts
- Accepted norms in the systems engineering community that cannot be discarded
- Cohesiveness of requirements that define the function

E.3.2.4 Synthesize Alternative Solutions

The starting point for the derivation of alternatives was the prototype GSEP that was derived from several sources, including Blanchard and Fabrycky (1990) and Department of Defense (1994). The prototype GSEP was then modified to address all of the derived systems engineering process requirements. There were several historical solutions that are maintained.

E.3.2.5 Evaluate Alternatives

The trade-offs between the alternate GSEPs and other systems engineering processes were made during this activity. Each of the alternates was assessed and rationale given for the various weighting of the alternates. Each of the alternates was evaluated to determine how well it addressed the previously identified functions (verification) as well as how well it addressed the needs (validation). During this activity, the previously completed trade-offs were assessed to select the alternate that was chosen as the GSEP.

E.3.2.6 Validate and Verify Solution

One key element of developing any process model is risk associated with acceptance of the model by the audience in which it is intended. To be acceptable, any system must be validated and verified.

To validate the GSEP, it was determined that the process model must be based on an accepted standard and that it must be compliant with other sources of systems engineering knowledge. Of importance in this step of GSEP development was to stay clear of blindly accepting common thoughts on systems engineering. It was decided that there would be a three-pronged approach to validating the GSEP model.

- Start with the prototype GSEP that reflected the existing work in the field of systems engineering.
- Establish a formal review process that contained experts in Consortium technologies, Consortium representatives that were coordinating with ongoing systems engineering activities, and internal and external experts in systems engineering and its application. This formal review process would ensure that the GSEP was in alignment with verified Consortium process technologies and ongoing systems engineering activities. An external review would aid in the validation of the model by applying credibility to the review process. This credibility would exist due to the fact that the external reviewers would not have a stake in Consortium technologies and would be well known in the systems engineering community.
- Use the Consortium TAG as a source for grass-roots knowledge of the state of the systems engineering practice. The TAG would have the opportunity to review the GSEP during its development and provide feedback on its merits and shortcomings.

Verifying the GSEP was be a more difficult problem. True verification is achieved through tests or, in the case of processes, through demonstration and application. The GSEP team decided that there were three acceptable approaches to verifying the GSEP.

- Use the GSEP, or portions of it, in case studies or examples that relate to the needs of the audience. As noted earlier, the final GSEP model was developed using the GSEP prototype. This approach provided the GSEP team the opportunity to instantiate the tailored GSEP and to think through the process. This was done at a very high level in the instantiation of the GSEP to support engineering of product families (see Section 5).
- Review organizational implementations of systems engineering. To verify that the GSEP was applicable within different environments, such as DoD, civil government, and commercial markets, and to ensure that it was applicable to various phases of a system's life cycle, there needed to be a comprehensive review of the application of systems engineering across the spectrum of use. Due to development time constraints, this activity has been deferred as potential future work.
- Work with organizations in applying GSEP to their systems development efforts. Feedback from this work would give the Consortium a true verification of GSEP and would be essential for GSEP guidebook development.

During this activity, the GSEP team created compliance matrices that show how the GSEP satisfies 499B and the SE-CMM. This validates the GSEP against two different standards and proposed systems engineering process instances.

E.4 PROCESS MODELING APPROACH

One of the first activities undertaken by the Consortium's GSEP development team was to determine and establish a systems engineering process modeling environment. Of concern were methods and

tools that would permit the GSEP team to dynamically alter an evolving process model and store the critical information that described it. The team was also concerned that the models created would be easily communicated to systems engineers.

The team first determined the type of process modeling that would be conducted. The alternatives were to model the systems engineering process from a functional, behavioral, organizational, or informational perspective or a combination of these perspectives. The team was interested in both a functional and an informational perspective, but decided that the foundation of the GSEP for the initial technical report would be the functional perspective. Therefore, a functional modeling approach was selected. This modeling approach permitted the team to identify the primary systems engineering activities and the key information that flowed between those activities. The tool selected for the functional model had to be relatively easy to learn, from a user's perspective, and easy to understand, from a reader's perspective.

Secondly, the team focused on a method to capture information that described the process, in terms of entrance and exit criteria, inputs and outputs, issues, etc. For an answer to this problem, the team consulted the Consortium's *Process Definition and Modeling Guidebook* (Software Productivity Consortium 1994b), which advocates the use of a database to record process information in conjunction with a graphical modeling environment.

E.4.1 IDEF Modeling

The process modeling environment chosen was Design IDEF 3.1 for Windows, by Meta Software, which supports IDEF0 and IDEF1/1X modeling. The Draft Process Technologies Method and Tool Report (Software Technology Support Center 1993) stated, "The learning curve for readers of IDEF0 models is not steep; training requirements are minimal." IDEF is also a modeling approach that has been used by systems engineers in some Consortium-affiliated companies on some large projects. The problem with the tool was that there was a large amount of process data that could not be adequately captured (see Section E.4.2).

E.4.2 Process Database

To capture information that defined the GSEP and its development, the GSEP team selected a relational database, Paradox for Windows. The database scheme (see Figure 59) is based on the *Process Definition and Modeling Guidebook* and includes tables for capturing:

- Process activities
- Risks of tailoring the process activities
- Tool classes that support the process activities
- Tools that are members of the tool classes
- Roles that support the activities
- Requirements satisfied by the activities
- Resources that are needed to complete the activities
- Work products used in and produced by the activities

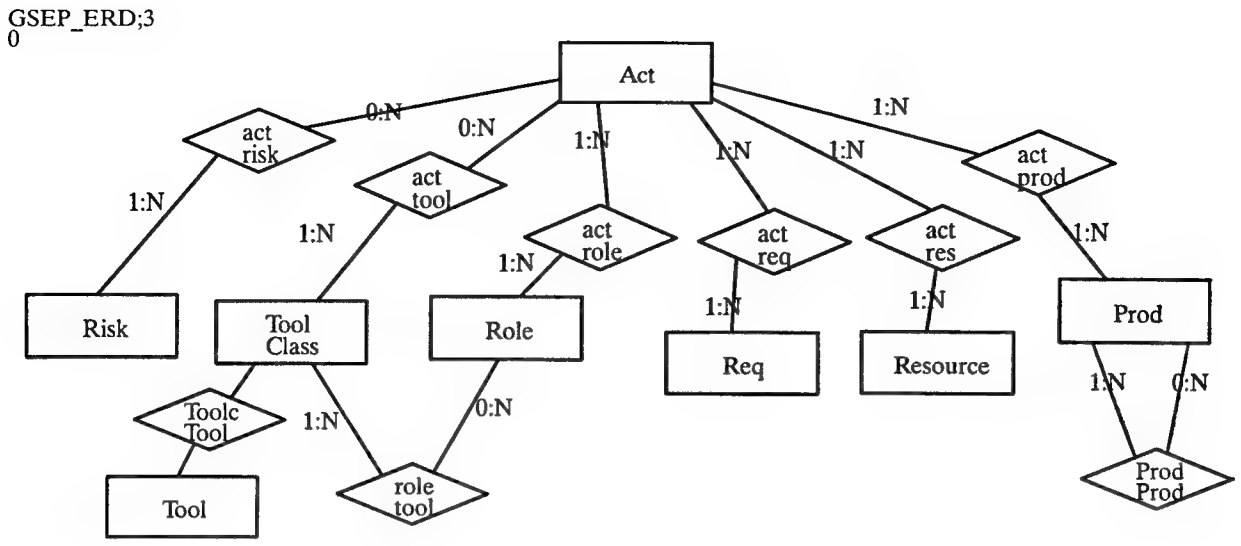


Figure 59. GSEP Database Schema

Links were established between these various tables in order to provide traceability and to show relationships. For instance, work products that are used by various activities, link the activities to each other. In this case, the "state" of the work product (e.g., preliminary, approved, baselined, etc.) and whether it is input or output are maintained in the relationship between the activity and the work product.

E.4.3 Interleaf

In order to publish this technical report, a desktop publishing tool was needed. Microsoft Word was considered, but the inclusion of the IDEF diagrams was an issue since Design IDEF was not OLE-compliant. This meant that when changes were made to the diagrams, all of the modified diagrams would have to be cut and pasted. Also, Word did not support the concept of shared paragraphs between documents. Interleaf was chosen because it supported linking to diagrams produced by IDEF, and it also supported shared components that were used to describe activities in the high-level GSEP description in Section 3 and in the detailed GSEP description in Appendix B.

Initially, the plan was to use Interleaf for the parts of the document that were not contained in Paradox, and Design IDEF. As the project progressed, there were many issues with this strategy that surfaced. First, the autoreferencing to sections, figures, and tables would not be supported by the tools. Second, Paradox did not support spell checking of the entire database and did not allow editing of a view of the database that was desired to produce Section 3 (since a query was involved that generated a report rather than providing a view into the database). As a result, a decision was made to convert all of the work products into Interleaf (except the IDEF diagrams) in order to produce this report. An unfortunate side effect of this decision is that the updates to the technical report from the final reviews were made to the Interleaf document, and the database no longer matches the report.

This page intentionally left blank.

APPENDIX F. GSEP AUTOMATED SUPPORT ENVIRONMENT

F.1 INTRODUCTION

This appendix summarizes the work done by the Consortium during 1994 on systems engineering tool support for the GSEP. The primary focus of this research is on defining tool requirements for a GSEP support environment and identifying COTS software that addresses the requirements. The identification of specific hardware to support GSEP has not been a part of this task. For the purposes of this appendix, all references to *tools* should be understood to mean *software applications*.

The work to date has included gathering information on specific tools, making contact with people actively working in the tools area, and sorting out the current issues and state-of-practice. The GSEP development team has developed top-level requirements for the tools, an initial tool classification scheme (tool classes), and identified some candidate tools. It should be emphasized that tool support is an ongoing concern. As new tools are developed and new needs surface, the tool support strategy needs to be evolved.

This tool investigation centered on gathering information that will allow the Consortium to address tool support issues and develop a foundation for future work in methods and tools; it did not emphasize the rigorous investigation and evaluation necessary to make final tool selections. The tools identified are only representative and may not meet all requirements for a particular tool class.

The tools mentioned in this report are not specifically recommended or evaluated, either among themselves or relative to other commercially available tools that are not mentioned. Furthermore, the accuracy and completeness of individual tool descriptions have not been validated with the vendors offering these tools and should not be considered authoritative. For such information, the respective vendors should be contacted directly.

F.1.1 Objectives and Approach

The primary goal of the work this year was to lay a foundation for future work in systems engineering methods and tools. The GSEP development team began by developing a preliminary set of automation support requirements based on the activities of the GSEP. The team then started identifying known tool experts, speaking with them and getting suggestions for other people to contact and other sources of information (see Section F.5 for a list of references). From this research, the team developed a tools list (see Section F.6) and contacted vendors for information.

Once familiarity with the tools was obtained, the tool classification scheme was developed (see Section F.4.1). Upon completion of the tool classification scheme, a matrix that maps tool classes to

GSEP activities was developed (see Table 5). Finally, the candidate tools were divided into tool classes providing an indirect mapping from tool to GSEP activity. Figure 60 graphically depicts the relationship between the individual tools, the classes which include the tools, and the GSEP activities which are supported by the tool classes. Because of the broad functionality of some of the tools identified, it is possible for a tool to fit into more than one tool class, and each tool class may include many tools. Similarly, many of the tool classes support more than one GSEP activity, and each GSEP activity is supported by one or more tool classes. This cardinality (many to many) is also shown in Figure 60.

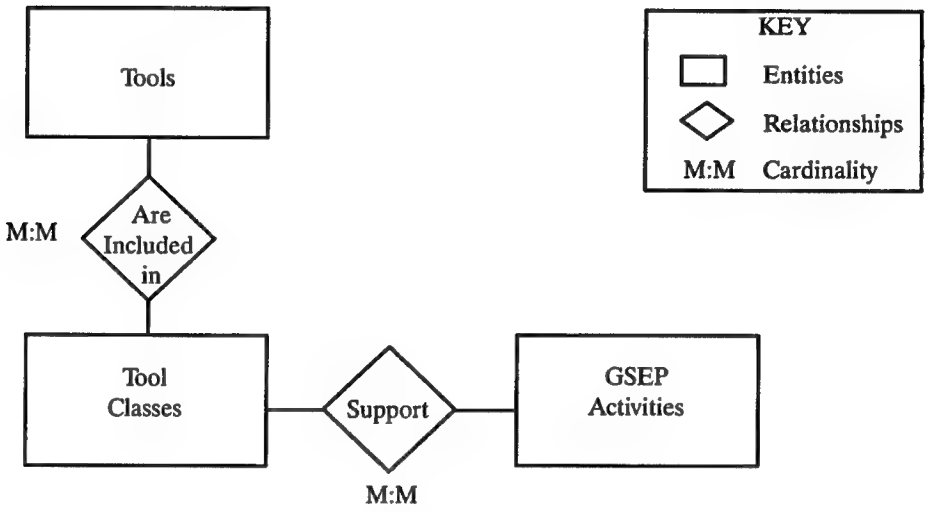


Figure 60. Strategy for Defining Tool Support

F.1.2 State-of-Practice

Currently, systems engineering is transitioning from a highly manual intensive effort to a more automated process. This is recognized as being essential to achieving gains in productivity necessary to remain competitive in today's dynamic marketplace. There are hundreds of automated tools available today which help the systems engineer perform the many tasks associated with the systems engineering process. However, there is no one tool that will satisfy all of a systems engineer's requirements, nor is there a set of integrated tools which comes close to meeting all the typical needs of a systems engineer.

Although the current state of systems engineering automation is immature, three conditions exist which indicate that it is an historically appropriate time for improvement. These conditions, as referenced in Kuhn (1993), are:

- The systems engineering process is being defined and baselined.
- Systems are becoming extremely complex.
- Desktop computing has become powerful and widespread.

F.2 AN INTEGRATED TOOL SUPPORT ENVIRONMENT

The automation of the systems engineering process cannot be fully achieved without an integrated computing environment. At the highest level, this environment would provide a set of software tools

that work together and share data, as well as a common repository for project documentation and data. The members of a systems engineering team are numerous, their positions in the company are varied, and more often than not, they are located in geographically distant areas. For these reasons, there will most likely be many different computing platforms that must be able to communicate over some distance. Therefore, the environment and all its applications must operate on a variety of different hardware platforms and operating systems, all of which are members of a local and/or wide area network.

F.2.1 General Tool Support Requirements

The objectives of an integrated environment are to improve the productivity of the systems engineering team and the quality of the system produced. Figure 61 shows a template that can be used for deriving top-level requirements for an automated GSEP support environment. The requirements include functional, interface compatibility, terminal/network, support services, and cost. During the process tailoring activity referred to in Section 4, each of these requirements is specified for the particular application.

It is easy to envision such an environment and the productivity improvements that automating GSEP would offer. Among the features provided might be an enactable process model for automation and training of a systems engineering process defined from the GSEP; a common repository, or at least the appearance of a common repository, for storing, searching, and retrieving all project data; provisions for data security, backup, archiving, and retrieval; a means for electronic communications; configuration management capability; a word processor or desktop publishing system with traceability to data; a requirements management tool with traceability in and among requirements as well as externally to other applications; modeling and simulation tools for describing a wide variety of systems; and a host of other speciality applications all of which are capable of sharing data (see Figure 61).

Although an integrated systems engineering environment, such as the one just described, is far from becoming a reality, some strides are being made. The NCOSE has a working group doing research into the issues involved in integrating tools. Also, Stephen Andriole of Drexel University has created a prototype of a systems engineering environment using a Microsoft Visual Basic interface to tie various COTS tools together in a Microsoft Windows environment. It will take a cooperative effort between vendors, academia, users, and user groups to create the first truly integrated systems engineering environment.

F.2.2 Management Process Tool Support Requirements

Documentation tools play a very important role in automating the GSEP management process. They can be used to create and maintain the project plans, as well as produce work products such as the work breakdown schedule (WBS) and the SOW. Many of the scheduling tools can also be used to create these work products. Planning and scheduling tools have the added advantage of providing differing views of the product status (e.g., graphical) and supporting status and progress reviews. The ideal situation is an integration between a scheduling tool and a documentation tool. This would allow for managing and tracking a project's progress and then automatically producing documentation for status reviews. To some extent, a number of tools offer this integrated capability today.

1.1. Functional Requirements
- 1.1.1. Management Process Support
- 1.1.2. Technical Process Support

1.2. Interface Compatibility Requirements
- 1.2.1. Platform Compatibility
- 1.2.2. Multiuser Integration
- 1.2.3. Supported Input/Output Devices
- 1.2.4. Communications
- 1.2.5. User Interface

1.3. Platform/Network Requirements
- 1.3.1. Availability, Accessibility, and Portability
- 1.3.2. Response Time
- 1.3.3. Storage Capacity
- 1.3.4. Access Control
- 1.3.5. Version/Baseline Control
- 1.3.6. Classified Data

1.4. Support Services
- 1.4.1. Data Backup, Archiving, and Retrieval
- 1.4.2. Installation and Repair
- 1.4.3. Network Management
- 1.4.4. Tool Configuration Management
- 1.4.5. Training

1.5. Cost
- 1.5.1. Acquisition (Per Seat)
- 1.5.2. Training
- 1.5.3. Maintenance and Support

Figure 61. GSEP Automated Support Environment General Requirements

Integrated tool support for the GSEP management process would include integration of tools such as a project management tool (for scheduling, resource planning, and project planning), a costing tool (for estimating project life-cycle costs), and a documentation tool (for printing the WBS). The system development plan is likely created using documentation tools in combination with outputs from many of the tools mentioned in the previous paragraph. Therefore, it is critical that the documentation tool be able to import text and graphics in a variety of formats.

F.2.3 Technical Process Tool Support Requirements

Requirements management tools play a major role in the GSEP technical process. These tools assist in capturing, organizing, and providing traceability of system requirements. In most cases, these tools provide links (typically custom) to modeling and simulation tools for requirements analysis and system design, thus establishing traceability from requirements to design. Modeling tools with their graphical user interfaces and hierarchical decomposition, provide a convenient means for conceptualizing the system solution.

Once the system is acceptably modeled, the speciality disciplines use simulation tools to analyze the model from their perspective (e.g., for performance, reliability, maintainability, or security). The results of the system analysis (including results from simulation and domain-specific design analysis) are often used by decision analysis tools to perform trade studies. In some circumstances, it is desirable

to feed information from specific design analysis tools back into the modeling tool for fine tuning of the system design. Finally, documentation tools are necessary for documenting the results and, in many cases, managing the volumes of project documentation. Again, the ability to import outputs from other tools (e.g., high-level model diagrams) into project documentation is essential.

F.3 INTEGRATION PROBLEMS AND SOLUTIONS

F.3.1 Problems and Issues

There are many reasons why an integrated environment is not yet a reality, and primary among these is the issue of standards. For applications to communicate, they must have a common language. It was not until recently that standards for data interchange, such as Standard Generalized Markup Language (SGML) for text and Computer Graphics Metafile (CGM) for graphics, have begun to emerge. However, defining a common data type is only the beginning. In some cases, the data can be exchanged in ASCII, but the organization of the data must be communicated. The receiving program needs a way to understand exactly what information it is importing. Unless the vendor has explicitly provided a way to import data from a particular tool, the import facility must be flexible enough to allow the user to define input fields, or an application programming interface must be provided.

Another factor that has impeded the progress toward integration is that, historically, vendors have kept their systems proprietary to protect their investment. However, without a published interface, it is impossible for any kind of integration to take place. More recently, vendors have come to realize that publishing interface information, as well as conforming to standards, makes their applications more appealing in terms of integration, and that this translates into increased sales. This issue is somewhat complicated by the use of in-house proprietary applications for which outside vendors could not possibly provide integrations.

Unfortunately, there are other nontechnical issues which, in some cases, have made integration impossible. When working on classified or high security government projects, regulations may dictate that certain systems remain isolated. Also, the fear of proprietary data leakage has kept many systems from joining a network.

F.3.2 Solutions

As the lack of tool integration becomes more of an issue, many tool vendors are offering custom links to specific applications to provide integration. A custom link allows data to be transferred from one program to another, obviating the need to reenter the data and, in some cases, providing traceability between tools. While this solution may be appropriate in some instances, it certainly is not the final answer. For example, the vast number of speciality analysis programs makes it unreasonable to consider that custom solutions could be provided in all cases. There are other problems inherent to custom solutions as well. When new versions of tools are released, custom interfaces often have to be modified to support new internal data structures. Also, if the need to replace a tool arises (e.g., the tool vendor is out of business or no longer supports the tool), then the custom application must be completely reengineered to address the replacement tool. This information supports the thesis that custom integrations are not the answer. The solution to tool integration will come from published interfaces, standard data interchange, and programmable application interfaces.

A new and promising technology is Object Linking and Embedding (OLE). This is Microsoft's answer to tool integration. OLE allows objects (charts, graphs, documents, sound, video, etc.) from any

compliant program to be embedded in another compliant program. For example, this would allow a document (work product) embedded in an engineering tool, to be updated in the engineering tool, yet published from the documentation tool. Microsoft is encouraging vendors to create OLE-compliant software by publishing the specifications. The major drawback with this technology is that it is platform specific. A similar technology that would support heterogeneous platforms is needed.

F.4 TOOL SELECTION

The environment described in Section F.2 is the backbone for a GSEP-automated support environment. Implementation of this environment with a suite of COTS tools would provide specific integrated support for GSEP. Although an integrated environment has not yet been implemented, there are numerous COTS tools available today that automate the many functions of the GSEP management and technical processes. These tools range from simple documentation tools, standalone spreadsheets, and decision analysis tools to very complex requirements management and simulation applications. Making the correct tool selection is a complex and time-consuming task.

The integration of activities in the GSEP adds a level of complexity to tool selection. If selected tools are to effectively support the integrated activity set, they must either be interoperable or the user must manually provide the data exchange between tools. Manual data exchange is time-consuming and error prone and is typically a last resort that is only considered if no other alternative exists. Because tool interoperability is part of the selection criteria, it must be incorporated into the selection process.

The GSEP tool selection process iterates between the following two considerations:

- Deciding which is the most appropriate tool(s) for each activity
- Deciding how the individual tools can be used together effectively

First, candidate tools are selected for each of the activities. There are many criteria that need to be considered when selecting candidate tools, such as how well the tool satisfies the formulated requirements, as well as tool capability, cost, performance, usability, availability, and familiarity. Then, interoperability issues are used to select a couple of reasonable tool choices for each activity from the initial set of candidate tools. The selection process iterates between choosing the most appropriate tool for the activity and evaluating how the choices fit together until commitment is reached on a final set of support tools.

During the selection process, the initial emphasis on selecting the most appropriate tool for each activity shifts to selecting the best suite of tools for the activities that will be automated. An integral part of selecting the tool suite is understanding how the tool interfaces will be accomplished. Tool selection is typically followed by a tool implementation plan that maps tools to supported activity functions and describes how tool interoperability will be handled.

The GSEP can be tailored for use as the GSEP tool selection process. Using a tailored GSEP for tool selection provides a well-defined, detailed approach that considers the tool application context, tool requirements, selection risks, and participant commitment in making tool selections. However, it is not necessary to engineer an elaborate process to make tool selections. If the tool selection is part of establishing a systems engineering environment that will significantly impact the engineering efforts of the organization, then there is justification for creating a tailored GSEP that provides a rigorous selection process. If, on the other hand, the selection is to decide between a series of in-house tools

to use on one small or moderate size engineering effort, less rigor is needed in the selection process, and the tailored GSEP should reflect the reduced scope. Certainly, the effort invested in tool selection should be proportional to the expected benefit from the tool usage.

F.4.1 Tool Classification

Because of the enormous number of tools available, the GSEP development team has classified tools so that it is possible to compare like tools and make informed tool selections. For the purposes of support for GSEP, tools have been grouped in very general tool classes. Certainly, the tool classes presented here may not be the best set, or even a complete set; the team expects this classification scheme to grow and mature over time. The following list describes the various tool classes with short explanations of their top-level requirement.

Note: When the general requirements for the tool classes were being defined, it became evident that there were some requirements that are common to all classes. To avoid repetition, these requirements are not enumerated with each class. First, all tools are expected to have a graphical user interface. Although there are a few tools that have yet to be ported to a standard windowing environment, most will have such a version in the near future. Second, it is assumed that any modeling or visualization tool will support hierarchical decomposition. Third, all tools should be able to import and export data in standard formats.

- ***Configuration Management.*** Configuration management tools help the system engineer manage project data by providing a common repository with a user-friendly interface. These tools must support heterogeneous platforms, track and record changes, have a security mechanism and an easy-to-use check in and check out mechanism, and provide support for different object types (e.g., graphics, text, video, audio, drawings, and schedules). An optimum configuration management tool would provide for traceability of attributes on requirements and specifications.

- ***Costing.*** Costing tools provide cost estimation based on historical data, skill levels within a domain, labor rates, and other hard costs (e.g., materials). These tools should support the rollup of costs into a higher level structure that defines anticipated costs for milestones in the system development. These tools should also support ranges of costs along with belief and confidence for the ranges.

- ***Decision Analysis.*** Decision analysis tools help the system engineer evaluate complex alternatives given uncertainty, value preferences, and risk preference. These tools must provide support for quantitative as well as qualitative data, support multicriteria evaluation, and provide a variety of methods for describing uncertainty.

- ***Documentation.*** Documentation tools provide support for creating, maintaining, and publishing all systems engineering documentation. Capabilities include generation of tables, index generation, autonumbering, support for scientific notation, graphics, audio, and video. The ability to electronically search, select, and view data and documentation should be provided.

- ***Engineering Data Management.*** Engineering data management tools allow the system engineer quick and easy access to all engineering data. These tools should support searching, selection, and viewing of all kinds of engineering data (drawings, references, documentation, schedules, etc.).

- ***Forecasting.*** Forecasting tools assist system engineers in deciding future events based on historical data. These tools are supported by various methodologies such as hierarchical influence diagramming, exponential smoothing, and decision tree logic.

- ***Modeling (Static).*** Modeling tools help the system engineer visualize the system from different perspectives. These tools should allow a systems engineer to graphically describe a system solution and analyze different aspects of the solution. Additionally, a graphical editor should make the creation of models quick and easy. Other capabilities must include, verification and validation, sensitivity analysis, completeness and consistency checking, and a mechanism for providing traceability to requirements.

- ***Quality Management.*** Quality management tools employ the Quality Function Deployment (QFD) methodology to assist in determining rankings. These tools use a "house of quality" matrix to depict relationships (i.e., between customer desires and product design parameters).

- ***Requirements Management.*** Requirements management tools help system engineers capture, visualize, and manage requirements. These tools should support traceability between requirements, provide links between functional and performance requirements, provide automatic document generation, support configuration management of requirements, and support linkages to other tools for design, development, and testing.

- ***Risk Analysis.*** Risk analysis tools use simulation to define the uncertainty in any variable. These tools should support Monte Carlo or Latin Hypercube sampling, provide statistical methods support, define critical paths, and provide high resolution graphical output.

- ***Project Management.*** Project management tools help organize and account for all the tasks and resources that are needed to reach the objectives of a project. These tools often provide facilities for scheduling, planning, estimating, and metrics.

- ***System Simulation (Dynamic Modeling).*** System simulation tools support modeling and performance analysis of systems. These tools should allow the system engineer to perform a variety of simulations including data flow, functions, behavior, Petri nets, and user scenarios. Standard libraries should provide icons and modules for quick and easy system descriptions, and the user should be able to create custom libraries to facilitate reuse. Other capabilities must include support for mathematical and statistical methods, real-time interaction, verification and validation of models, links to outside data sources, and a mechanism for providing traceability to requirements.

- ***Test and Data Analysis.*** Testing tools provide support for verification and validation. These tools should assist in automating test plan generation and test data generation. Testing tools should support coverage analysis, test case management, regression testing, and capture and playback. Data analysis tools assist in reducing and analyzing the test data and provide statistical analysis.

Table 5 describes a mapping of the tool classes to the top-level GSEP activities, either management or technical activities. An X indicates that the particular tool class can be used in support of the corresponding GSEP activity. For example, configuration management occurs on many work products; however, there is a specific activity that takes the work products and performs the configuration management. Section F.6 provides a listing of candidate tools by class.

Table 5. Tool Support for Top-Level GSEP Activities

Tool Classes \ GSEP Activities	Understand Context	Analyze Risk	Plan Increment Development	Track Increment Development	Develop System Plan	Analyze Needs	Define Requirements	Define Functional Architecture	Synthesize Allocated Architecture	Evaluate Alternatives	Validate and Verify Solution	Control Technical Baseline
Configuration Management			X		X							X
Costing	X	X	X	X	X				X	X		
Decision Analysis	X	X	X	X	X			X	X	X		
Documentation	X	X	X	X	X	X	X	X	X	X	X	X
Engineering Data Management	X		X		X	X	X	X	X	X		X
Forecasting	X	X	X		X	X				X		
Modeling (Static)						X	X			X		
Quality Management	X	X	X		X	X	X	X	X	X	X	X
Requirements Management	X			X			X	X	X		X	
Risk Analysis		X	X		X	X				X		
Project Management		X	X	X	X							
System Simulation (Dynamic Modeling)						X	X	X	X			
Test and Data Analysis				X							X	
	Management					Technical						

F.5 SOURCES OF INFORMATION

The following references are sources of information on systems engineering tools. Each source is followed by an abstract of its contents. They are provided here as information sources for further research on this topic. Due to the volatile nature of the software industry, specific information on tools and vendors may have changed since this publication.

- *Application of the Systems Engineering Process to Define Requirements for Computer-Based Design Tools*, edited by Benjamin S. Blanchard, Jr., Wolter J. Fabrycky, and Dinesh Verma, and published in March 1994 by The Society of Logistics Engineers, 8100 Professional Place, Suite 211, New Carrollton, Maryland 20785.

  This report defines a generic systems engineering process and then uses this process as a framework for classifying tools. System activities are described and applicable tools and models are related to the activities. A survey with descriptions and vendor information on over 250 selected tools is included as an appendix.

- *Center for Multidisciplinary Information Systems Engineering Commercial Off-the-Shelf Requirements, Prototyping, and Software Engineering Applications Software Database,* produced by the Center for Multidisciplinary Information Systems Engineering at Drexel University College of Information Studies, 32nd and Chestnut Streets, Philadelphia, Pennsylvania 19104.

  This 4th Dimension database contains information on over 300 systems engineering tools. For each tool, information is included on the vendor, platform, operating system, life-cycle phase, user experience, capabilities, price, memory, and display. Searches may be constructed using a systems engineering life cycle, user experience, operating system, platform, price range, and capability. A search editor for defining custom searches is also available.

- *Guest Lecture Documentary Series, Computer-Aided Systems Engineering*, presented by Dr. Stephen J. Andriole (Software Productivity Consortium 1994a)

  Dr. Stephen Andriole is a noted systems engineering tools expert. He is a professor at Drexel University and director of the Center for Multidisciplinary Information Systems Engineering. This is a video tape of a presentation he gave in the spring of 1994 at the Consortium. He presents a systems engineering process and then explains how inexpensive COTS tools can be used to automate many systems engineering tasks.

- *Automating Systems Engineering* by Dorothy A. Kuhn, a research project presented to the faculty of The University of Texas at Dallas in partial fulfillment of the requirements for the degree of Master of Arts in Interdisciplinary Studies, November 1993.

  Dorothy Kuhn is well known in the systems engineering arena. She is very knowledgeable about tools and is actively working on the Systems Engineering Capability Maturity Model. Her report examines the state of systems engineering automation and what is needed to effectively automate the systems engineering process. The report also includes evaluations of several tools with pointers to more information.

- *An Integrated Approach to Complex Systems Analysis and Performance Simulation* by Kenneth J. Peterson, and Brian W. Mar. This article was published in the Proceedings of the Fourth Annual International Symposium of the National Council on Systems Engineering, Volume 1, August 10-12, 1994, page 213.

  This article discusses simulation and its support for the systems engineering process. It presents a classification of simulation types and then maps them to systems engineering tasks. Information is also presented on defining requirements for choosing a simulation tool.

- *Requirements Analysis and Design Technology Report March 1994* by the Software Technology Support Center at Ogden ALC/TISE, 7278 4th Street, Hill AFB, UT 84056-5205, 1-801-777-7703.

  This report is a synopsis of the work done at the Software Technology Support Center in evaluating requirements analysis and preliminary design computer-aided software engineering products. It contains an extensive listing of software which includes product information, contact information, intended customers, information on underlying methodology, which life-cycle phase and activity are supported, and a description of the product. Although this report primarily targets software engineering, many of the tools listed are also applicable to systems engineering.

- *Test Preparation, Execution, & Evaluation Software Technologies Report February 1993,* by the Software Technology Support Center at Ogden ALC/TISE, 7278 4th Street, Hill AFB, UT 84056-5205, 1-801-777-7703.

  This report is a synopsis of the work done at the Software Technology Support Center in understanding software testing technologies. It contains an extensive listing of software which includes product information, vendor information, information on the underlying methodology, product tool classifications, and a description of the product. Although this report primarily targets software engineering, many of the tools listed are also applicable to systems engineering.

- *Documentation Technology Report April 1994,* by the Software Technology Support Center at Ogden ALC/TISE, 7278 4th Street, Hill AFB, UT 84056-5205, 1-801-777-7703.

  This report is a synopsis of the work done at the Software Technology Support Center in understanding documentation technology. This report delves into the technology of documentation and document management. It describes how effective documentation and documentation management can improve productivity. A good amount of information on current products is included.

- *Software Configuration Management Technology Report September 1994,* by the Software Technology Support Center at Ogden ALC/TISE, 7278 4th Street, Hill AFB, UT 84056-5205, 1-801-777-7703.

  This report addresses issues of software configuration management, specifically those issues that concern the Air Force agencies developing embedded software. It presents an overview of software configuration management technology and provides references for further information. Also included is an extensive list of software configuration management tools with information on the vendor, the product, pricing, and platform requirements. Each entry also has a short product description.

- *National Instruments Solutions*, April 1994. National Instruments Corporation, 6504 Bridge Point Parkway, Austin, TX 78730-5039, 1-512-794-0100.

  This is a directory of third party products and consultants for LabVIEW and LabWindows. It contains complete product and vendor information for over 250 products.

F.6 CANDIDATE TOOLS

Table 6 contains a list of some of the more popular systems engineering tools. Note that all information is directly from the vendors, the Consortium did not perform any hands-on evaluations this year. The information provided should be enough to spark interest in a particular tool. At that point, it is necessary to do a more complete evaluation to ensure that the tool meets all of the defined requirements. All tools have been grouped into categories according to their major function. In many cases, add-on modules (optional tool extensions) provide functionality appropriate to different classes. Most of the time, however, this add-on functionality is only appropriate within that particular tool and does not have general applicability.

Table 6. Systems Engineering Tools

Configuration Management		
ClearCase	Atria Software, Inc. 24 Prime Park Way Natick, MA 01760 1-508-650-5100 1-508-650-3573 (Fax)	This is a configuration management tool for heterogeneous UNIX development environments. It has a user-customizable graphical user interface and implements version control, process control, build management, and workspace management. Platform: Most UNIX
Costing		
Life Cycle Cost Calculator (LCCC)	Systems Engineering Design Laboratory 146 Whittemore Hall College of Engineering Virginia Tech Blacksburg, VA 24061	"The LCCC model aids in the accomplishment of a life-cycle cost analysis, using a cost breakdown structure (CBS) methodology. Objectives and activities are linked to resources, and constitute a logical breakdown of cost by functional activity area, major element of a system, and/or one or more discrete classes of common or like items. It provides a mechanism for the initial allocation of cost, cost categorization and finally, cost summation. It has the capability of identifying cost contributions, in both real and discounted dollars, at any level in the CBS" (Blanchard, Fabrycky, and Verma 1994). Platform: PC/DOS
Decision Analysis		
Decision Analysis Techniques	Lionheart Press, Inc. P.O. Box 20756 Mesa, AZ 85277-0756 1-602-396-0899	This includes a specialized book and set of programs for decision analysis. The book covers decision theory. Six programs are included: TREE (for the analysis of a sorted decision tree), DTABLE (decision tables using utility, monetary or opportunity loss criteria), BAYES (calculation of posterior probability), AHP (gives eigenvalues of reciprocal matrices), ELECTRE (decision making), and UTILITY (calculation of utility probabilities). Platform: PC/DOS, Macintosh

Table 6, continued

Decision Analysis (continued)		
Demos	Lumina Decision Systems 4984 El Camino Real, Suite 105 Los Altos, CA 94022 1-415-254-0189	Demos is a hierarchical decision modeling system. Its graphical interface allows the user to create and analyze probalistic models for risk, cost, and policy analysis. The integration of hypertext and model is designed to support communication within a group. It supports influence diagramming, a powerful forecasting methodology. Platform: Macintosh
Expert Choice	Expert Choice, Inc. 4922 Ellsworth Ave. Pittsburg, PA 15213 1-412-682-3844 1-412-682-7008	Expert Choice is a decision support software package based on the analytic hierarchy process. It is a multicriteria decision-making product which helps the user structure problems, improve evaluations, prioritize alternatives, and perform what-if sensitivity analysis. Platform: PC/DOS, PC/Windows
Logical Decision	Logical Decisions 1014 Wood Lily Drive Golden, CO 80401 1-303-526-7536	Logical Decision is a graphical alternative and preference evaluation tool. It uses quantitative and qualitative variables to come up with an importance ranking. It provides what-if analysis and help screens. Platform: PC/Windows
SLIM	Quantitative Software Management, Inc. 2000 Corporate Ridge, Suite 900 McLean, VA 22102 1-703-790-0055	SLIM is an expert system for strategic planning for software projects. It allows the user to do estimating and economic analysis relating to design processes. The user provides assumptions and constraints, and SLIM presents an optimum solution. It also does trade-off analysis and risk analysis. Platform: PC/Windows
Documentation		
Interleaf	Interleaf Inc. Prospect Place 9 Hillside Ave. Waltham, MA 02154-9524 1-800-955-5323	Interleaf is a very powerful desktop publishing system. It supports work group publishing, revision tracking, documentation structure, and long document assembly and layout. It has a variety of filters for importing and exporting text and graphics including SGML and Computer-Aided Logistics Support (CALS). Add on packages provide hypertext, document management, and database links. Platform: Most UNIX, PC/DOS, PC/Windows(soon)

Table 6, continued

Documentation (continued)		
Microsoft Word	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399 1-206-882-8080	Microsoft Word is a powerful word processor. It supports page layout, tables, macros, merging, direct linking, templates, and style sheets. Platform: Macintosh, PC/Windows
Engineering Data Management		
Pangea	Telos Corporation 460 Herndon Parkway Herndon, VA 22070 1-800-444-4432	The Pangea Model bridges past, present, and future data requirements using a virtual database integration strategy. It is an environment for integrating information stored in a wide range of data structures: flat files, hierarchical databases, network databases, relational databases, and object-oriented databases. The graphical user interface allows users access to data transparently, regardless of its form or location. Platform: Server runs on most UNIX platforms; clients are available for many other machines.
Forecasting		
Forecast!	Intex Solutions, Inc. 35 Highland Circle Needham, MA 02194 1-617-449-6222 1-617-444-2318 (Fax)	This is a forecasting tool which is an add on for Lotus 1-2-3. It does time series analysis, multiple regression, and other statistics. It has a graphical user interface and produces charts and graphs. Platform: PC/Windows
STORM (STatistics Operations Research and Management)	Storm Software Inc. 24100 Chagrin Blvd. Cleveland, OH 44122-5535 1-216-464-1209 1-216-464-4222 (Fax)	STORM is menu-driven, quantitative modeling software which includes a variety of quantitative modeling tools. Some of the individual modules include linear programming, forecasting, statistics, integer programming, project management, distance and flow networks, queueing analysis, investment analysis, facility layout, and statistical process control. Platform: PC/DOS

Table 6, continued

Modeling		
CORE	Vitech Corporation 2422 Rocky Branch Road Vienna, VA 22181 1-703-883-2270	CORE provides requirements extraction and analysis, functional modeling, and analysis capability for modeling the physical architecture of a system. It has a single integrated systems design database to provide traceability and prevent inconsistencies. User modifiable report templates for automatic document generation are also included. Platform: PC/Windows
Design/IDEF	Meta Software Corporation 125 Cambridge Park Drive Cambridge, MA 02140 1-617-576-6920 1-800-227-4106	Design/IDEF is an IDEF modeling tool which helps the user stay within the IDEF paradigm. It supports IDEF0, IDEF1, IDEF1X, and entity relationship diagrams. It has a running data dictionary and a glossary and maintains activity-based costing information. The file format is compatible between platforms. Behavorial models can be exported to Design/CPN for simulation and system analysis. Platform: PC/Windows, most UNIX, Macintosh
ENVISION	Future Tech Systems, Inc. 824 East Main Street Auburn, WA 98002 1-206-939-7552	Envision is a graphical, multimedia, repository, based modeling system. It can be used to document processes, systems, and structures. The models can be simulated and animated with Future Tech's SIMVISION product. Other applications include activity-based costing, statistical analysis of workflow metrics, function point analysis, systems analysis and design, and data modeling. Platform: PC/Windows
MetaDesign	Meta Software Corporation 125 Cambridge Park Drive Cambridge, MA 02140 1-617-576-6920 1-800-227-4106	MetaDesign is a flexible hierarchical modeling tool. It allows the user to create general flows of objects, place and resize objects, and create hypertext from objects. Through the use of palettes, the user can select a variety of graphics or create their own. Platform: Macintosh, PC/Windows

Table 6, continued

Modeling (continued)		
SLATE	TD Technologies, Inc. 6140 Parkland Blvd., Suite 200 Mayfied Heights, OH 44124 1-216-460-4700 1-216-460-4705 (Fax)	SLATE is a requirements management tool and system-level development tool. Along with requirements management, it does technical budgeting, functional block diagrams, and automatic document generation. It allows multiple views of the design data, provides transitional mapping relationships (in addition to hierarchical), and allows for graphical design capture. It is a multiuser tool. Platform: Sun
Software through Pictures StP/IM StP/OMT StP/SE StP/T	IDE 595 Market Street, 10th floor San Francisco, CA 94105 1-800-888-4331 1-415-543-0900 1-415-543-0145 (Fax)	Software through Pictures is a suite of integrated analysis and design tools that helps to build quality software. There are four core technologies: • StP/SE for structured analysis and design. It supports the most popular structured methodologies. Its real-time extensions include control flow, state transition, and control specification diagrams. • StP/IM for information modeling to help in analyzing data. It assists in capturing the conceptual model of the data and then extracting a logical model from the conceptual model. • StP/OMT for object-oriented development and implementation, supporting the Object Modeling Technique. • StP/T for Testing (see Testing tool section). Platform: Sun, IBM RS6000, HP, DEC
System Architect	Mentor Graphics Corporation 8005 South West Boeckman Road Wilsonville, OR 97070 1-503-685-7000 1-301-990-1265	System Architect provides an easy structured approach to developing models. It has a data flow diagram graphical editor, a state transition diagram graphical editor, a state transition matrix/general table editor, and a state machine animator for logical verification of finite state machines. Platform: HP, Sun, DEC, IBM RS/6000, NEC

Table 6, continued

Modeling (continued)		
Teamwork **Team***work*/Ada **Team***work*/FORTRANRev **Team***work*/IM SQL **Team***work*/OOA **Team***work*/OOD for C++ **Team***work*/RT **Team***work*/SA **Team***work*/SD **Team***work*/SIM **Team***work*/TestCase **Team***work*/IPSE_toolkit DocTools for Interleaf	Cadre Technologies, Inc. 222 Richmond Street Providence, RI 02903 1-401-351-5950 1-401-351-7380 (Fax)	**Team***work* is a hierarchical modeling tool for design of real-time software systems. Additional capabilities offered through add-on modules include simulation, code generation, document generation, and testing. It consists of the following modules: • /Ada for Ada analysis and design. This module does reverse engineering and some Ada code generation. • /IM SQL, the information modeling extension for analyzing data and information relationships. Users can create SQL DDL and Chen data models. • /OOA, object-oriented analysis module which follows the Shlaer-Mellor method. • /OOD, the design module for Shlaer-Mellor method. Supports automatic C++ code generation. • /RT, realtime extension, for interactive development of models for event-driven, transaction-driven, and real-time systems. • /SA, structured analysis tool that follows Yourdon DeMarco method. It supports graphical requirements definition. • /SD, structured design extension. Allows the user to graphically design the layout of the project (before coding). • /SIM, the simulation extension supporting dynamic verification of real-time structured analysis and models. • /TestCase supports automatic generation of test cases. • DocTools for Interleaf, a package for creating documentation with the Interleaf desktop publishing system and for maintaining consistency between **team***work* objects and deliverable documents. • /IPSE_tolkit, is an integration toolkit for integrating **team***work* with other tools. Platform: Sun HP, DEC, VAX, OS/2 SGI, RS6000

Table 6, continued

Project Management		
Microsoft Project	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399 1-206-882-8080	Microsoft Project is a project management application which combines scheduling, planning, and charting with powerful presentation tools. Features include the ability to schedule resources across multiple projects, easy to enter and update project information, multiple page print preview, spell check and file compatibility across platforms. Platform: PC/Windows, Macintosh
Project Workbench	Applied Business Technology Corporation 361 Broadway New York, NY 10013 1-212-219-8945 1-800-4-projec	Four separate integrated products for total project management. Project Workbench is the scheduling, tracking, and analysis module. Other modules are Project Bridge Modeler which provides project planning, profiling and estimating; Methods Architect which allows for customizing a development methodology; and Metrics Manager which is a measurement tool. Platform: PC/Windows
Project Bridge Modeler	Applied Business Technology Corporation 361 Broadway New York, NY 10013 1-212-219-8945 1-800-4-projec	Project Bridge Modeler contains four separate integrated products for total project management. It provides project planning, profiling, and estimating; Other modules are Project Workbench, which is the scheduling, tracking, and analysis module; Methods Architect, which allows for customizing a development methodology; and Metrics Manager, which is a measurement tool. Platform: PC/Windows
Quality Management		
QFD/Capture	ITI 5303 DuPont Circle Milford, OH 45150 1-513-576-3900	QFD/Capture is a tool for managing the data involved with QFD analysis. Benefits include easy data input, generation of matrix output, performing complex calculations, and assisting with analysis of data (sorting, graphing, etc.). Platform: PC/Windows, Macintosh

Table 6, continued

Requirements		
Document Director	Compliance Automation, Inc. 17629 El Camino Real, Suite 207 Houston, TX 77058 1-713-486-7817 1-713-486-0115	Document Director is a DOS-based requirements management tool. It contains two tools: ReqMgr which enables you to write, maintain, and control documents, and Browse, which gives all personnel online access to documents. A Windows version called Vital Link will be available soon. Platform: PC/DOS
DOORS	Zycad Corporation Two Fountain Square 11921 Freedom Drive, Suite 550 Reston, VA 22090 1-703-904-4360 1-703-834-6622 (Fax.)	DOORS provides object-oriented requirements management. Features include dynamic document generation, graphical views of requirements, linkage between functional and nonfunctional requirements, application programming interface, and links to other tools. Platform: Sun, PC/Windows
RTM	Marconi Systems Technology 4115 Pleasant Valley Road, Suite 100 Chantilly, VA 22021 1-703-263-1260 1-703-263-1533 (Fax.)	RTM is a requirements traceabiltiy and management product. It will manage all relationships between a variety of project objects such as code modules, test specifications, test results, documents, and hardware. Interfaces to many standard CASE tools are provided (**team***work*, Software through Pictures, FrameMaker, Interleaf). It has automated versioning which allows baselining of requirements or other data. Platform: Sun/Solaris, Vax/VMS, DecAlpha/OSF, HP, and IBM RS/6000
Risk Analysis		
@RISK	Palisade Corporation 31 Decker Road Newfield, NY 14867 1-800-432-7475	@RISK is a risk analysis and simulation add-on for Excel or Lotus 1-2-3. It provides all the necessary tools for setting up, executing, and viewing the results of risk analysis. It does Monte Carlo simulation. Platform: PC/Windows, Macintosh.
@RISK For Project	Palisade Corporation 31 Decker Road Newfield, NY 14867 1-800-432-7475	@RISK For Project is a risk analysis and simulation add-on for Microsoft Project. It provides all the necessary tools for setting up, executing, and viewing the results of project risk analysis. It utilizes Monte Carlo or Latin Hypercube sampling. Platform: PC/Windows, Macintosh

Table 6, continued

Risk Analysis (continued)		
Microsoft Excel	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399 1-206-882-8080	Microsoft Excel is a complete spreadsheet which has business graphics, a database, and powerful presentation tools. Some features include point and click to link multiple spreadsheets, extension macro functions, share files with other spreadsheets for Macintosh and PC, and context-sensitive help. Platform: Macintosh, PC/Windows
System Simulation		
BONeS (Block-Oriented Network Simulator)	The Alta Group of Cadence Design Systems, Inc. 919 East Hillsdale Blvd. Foster City, CA 94404 1-415-584-5800 1-415-358-3601	BONeS is used for developing, analyzing, and simulating communications or computer networks or system architectures. It has a graphical user interface, supports automatic C code generation, and comes with an application library of models. Platform: Sun, HP
Design/CPN	Meta Software Corporation 125 Cambridge Park Drive Cambridge, MA 02140 1-617-576-6920 1-800-227-4106	Design/CPN is a simulation tool based on a behavior-oriented methodology. The underlying formal principles are those of hierarchical Colored Petri nets (CP-nets). CP-net models are used to design and analyze real-time systems and transaction processing. Design/CPN accepts behavorial models directly from Design/IDEF. It also includes a graphical editor, an application generator, a reporting facility, and as animation capability. Platform: Macintosh, PC/Windows, Sun, HP
EASY5x	Boeing Computer Services P.O. Box 24346 MS 7L-46 Seattle, WA 98124 1-206-865-3302 1-800-426-1443	EASY5x is a tool used to model, simulate, analyze, and design large, complex, dynamic systems that are generally characterized by algebraic and/or differential equations. It is intended for scientists, engineers, and system analysts. It comes with a hydraulic library, an environmental control system library, and an air vehicle library. The graphical user interface makes it very easy to use. Platform: Mainframes, most UNIX

Table 6, continued

System Simulation (continued)		
Extend (Extend+BPR)	Imagine That, Inc. 6830 Via Del Oro, Suite 230 San Jose, CA 95119 1-408-365-0305	Extend is a high-level, object-oriented process modeling system. It combines a flexible, iconic modeling system (simulator) with a built-in language for extensibility. Features include hierarchical decomposition, animation, sensitivity analysis, statistical capabilities interruptible processes, and links to other tools. Extend+BPR allows the user to apply systems analysis techniques to process improvement efforts. A PC/Windows version will be out in December 1994. Platform: Macintosh
Foresight FS/Model FS/Sim FS/RQ-IF FS/Documentor FS/Coder$_c$ Bridegway	Nu Thena Systems, Inc. 1430 Spring Hill Road, Suite 220 McLean, VA 22102 1-703-356-5056	Foresight is a graphical, real-time, embedded systems modeling tool which allows modeling and simulation from initial requirements to detailed component architectures. It allows the codesign of embedded hardware and software. FS/Documentor works in conjunction with Foresight to provide automatic documentation generation. FS/RQ-IF establishes a dynamic link between Foresight and a variety of requirements traceability and information management tools. FS/Coder$_c$ is a C code generator which translates graphical Foresight system models into either ANSI or K&R C code for creating standalone executable models. Bridegway is a simulation blackplane that allows the linkage of applications over a network for creating distributed cooperating simulations and operational prototypes. Platform: Sun, HP
G2	Gensym Corporation 5180 Parkstone Drive, Suite 110 Chantilly, VA 22021 1-703-266-0203 1-703-266-9380 (Fax)	G2 is a real-time expert system. G2 and its layered application products provide a complete graphical environment for rapid prototyping of such things as real-time monitoring, diagnosis, quality management, decision support, and intelligent supervisory control. Some features include real-time concurrent operations; rule-based, procedural, and model-based reasoning; and real-time interfaces to databases for real-time data acquisition, analysis, and display. Platform: Most UNIX, WindowsNT

Table 6, continued

System Simulation (continued)		
iThink STELLA II	High Performance Systems, Inc. 45 Lyme Road, Suite 300 Hanover, NH 03784 1-603-643-9636	iThink is a high-level graphical modeling and simulation tool. It combines graphics, animation, and video to communicate ideas more effectively. Its multirun sensitivity analysis lets the user systematically vary any set of variables and then compile the results in comparative graphs or tables. iThink and STELLA are the same software, they come with different documentation. The documentation for iThink is geared toward business, whereas that for STELLA is focused more on academics or sciences. Platform: Macintosh and PC/Windows
MATRIXx SystemBuild Xmath AutoCode DocumentIt RealSim	Integrated Systems, Inc. 12030 Sunrise Valley Drive, Suite 300 Reston, VA 22091 1-703-391-6027	The MATRIXx product family of tools provide rapid design development, simulation, code generation, and automatic documentation generation. SystemBuild is a visual, intuitive block diagram modeling and simulation environment. Xmath is an object-oriented graphics and analysis package which provides interactive 2-D and 3-D color graphics. AutoCode automatically generates C and Ada real-time source code. DocumentIt automatically incorporates information from SystemBuild into written documents. RealSim provides for rapid prototyping, hardware-in-the-loop testing, and system simulation. Platform: Sun, HP
RDD-100 System Browser (RDD−100/SB) Requirements Editor (RDD−100/RE) Requirements Manager (RDD−100/RM) System Analyst (RDD−100/SA) System Designer (RDD−100/SD) Dynamic Verification Facility (RDD−100/DVF) Report Writer (RDD−100/RWE) Cadre Teamwork Interface (RDD−100/TWI) Interleaf Bridge (RDD−100/ILF)	Ascent Logic 2070 Chain Bridge Road, Suite 390 Vienna, VA 22182 1-703-827-9010	RDD-100 is an integrated environment for process modeling, product planning, and development. It provides full requirements analysis and traceability through behavioral modeling and simulation to component allocation. The System Browser is used to view the database. The Requirements Editor enables requirements to be captured and manipulated. The System Analyst allows the creation of system behavorial models. The System Designer includes all of the features of the entire product family as well as the ability to customize the user environment. The Dynamic Verification Facility provides behavorial simulation capability. The Report Writer is the printing interface, it provides predefined templates and the ability to create custom templates. The Cadre Teamwork Interface provides an output link to CDIF context and data-flow diagrams. The Interleaf Bridge works with the Report Writer to export information in Interleaf format. Platform: Most UNIX, PC/Windows, Macintosh

Table 6, continued

System Simulation (continued)		
SDS (System Design Station)	Mentor Graphics Corporation 8005 South West Boeckman Road Wilsonville, OR 97070 1-503-685-7000 1-301-990-1265	SDS provides automated system design from concept to production. It has a graphical user interface and provides hierarchical decomposition of models. It includes all the features of System Architect as well as providing for conceptual design and system simulation. Other features include requirements capture, rapid prototyping/test panel development environment, and interface to other tools. Platform: HP, Sun, DEC, IBM RS6000, NEC
SES/*workbench*	Scientific and Engineering Software, Inc. 4301 Wesbank Dr., Bldg. A Austin, Texas 78746 1-512-328-5544 1-512-327-6646 (Fax)	SES/*workbench* is an integrated system design and simulation environment with graphical interface. It can be used from the conceptual stage through functional and performance evaluation. Animation and extensive statistical analysis capabilities are provided. Platform: Sun, HP, DEC
SIMULINK (requires MATLAB)	The MathWorks, Inc. 24 Prime Park Way Natick MA 01760 1-508-653-1415 1-508-653-6284(Fax)	MATLAB is a technical computing environment and 4th generation language. It does numerical analysis and matrix manipulation. SIMULINK works in conjunction with MATLAB to provide nonlinear dynamical simulation of real-world systems. It provides graphical tools for building hierarchical block diagrams and provides the ability to build graphical interfaces. A C code generator is available for real-time embedded control, rapid prototyping, and simulation. There are other tool boxes available for speciality math applications such as signal processing and neural networks. The data is compatible across platforms. Platform: PC/Windows, UNIX, Convex, Cray, Macintosh
SIMVISION	Future Tech Systems, Inc. 824 East Main Street Auburn, WA 98002 1-206-939-7552	SIMVISION provides simulation and animation of Envision models. Platform: PC/Windows

Table 6, continued

System Simulation (continued)		
Statemate	i-Logix Inc. Three Riverside Drive Andover, MA 01810 1-508-682-2100 1-508-682-5995 (Fax)	Statemate is a modeling and simulation tool which is appropriate for modeling a wide range of applications at different levels. A system can be decomposed to functional units and the interfaces between the units can be specified. A statechart can be used to specify behavior; this can then be used to model complex behaviors graphically and concisely. Because it is based on a formal graphical language, the behaviors a user has specified can be simulated. Also, models can be checked statically for unusual behavior. It has a template driven document generator; allows for ASCII import from traceability tools; has a bidirectional CDIF interface; has the ability to do test panel generation; and can generate code in C, Ada, VHDL, and Verilog. Platform: Sun, HP, IBM RS6000, DECstation, VMS
TRANSCEND	TD Technologies, Inc. 6140 Parkland Blvd., Suite 200 Mayfied Heights, OH 44124 1-216-460-4700 1-216-460-4705 (Fax)	TRANSCEND is a high-level simulation environment that provides transaction level modeling. Its object-oriented technology gives users the ability to represent designs by data flow. It has complete global variable capability, as well as hierarchical design capabilities. It is very easy to capture a design at the graphical level without the need for detailed programming. It has a C++ programming language interface. Platform: Sun, HP
Testing		
BenchMaster	A&T Systems, Inc 12520 Prosperity Dr., Suite 300 Silver Spring, MD 20904 1-301-384-1425	BenchMaster is a multipurpose remote terminal emulator that simplifies conducting benchmarks. It can be used to measure the performance of a computer system by emulating the real-life workload. Platform: Most UNIX
StP/T (Software through Pictures Test Case Generator)	IDE 595 Market Street, 10th floor San Francisco, CA 94105 1-800-888-4331 1-415-543-0900 1-415-543-0145 (Fax)	StP/T automatically generates test cases based on specifications. It is integrated with StP analysis and design tools; test cases are generated from the analysis and design models which are stored in the StP repository. T can also run standalone on a PC running DOS, in which case it works from a specification matrix. Platform: Sun, IBM RS/6000, HP, DEC

Table 6, continued

Other		
Inspiration	Inspiration Software, Inc. P.O. Box 1629 Portland, OR 97207 1-503-245-9011	Inspiration is a visual idea development tool for planning and developing ideas from initial conception to final document. It allows you to create outlines, flow charts, idea maps, tree charts, presentation visuals, and proposals. Platform: Macintosh and PC/Windows
Labview	National Instruments 6504 Bridge Point Parkway Austin, TX 78730-5039 1-800-433-3488	Labview is an iconic programming language for simulation, analysis, and data acquisition. It includes libraries of virtual instruments. It allows data acquisition from outside sources, has many add-on toolkits, and can do real-time process control. Platform: Macintosh, Windows, Windows NT, Sun
Maple V	MathSoft, Inc. 101 Main Street Cambridge, MA 02142-1521 1-800-628-4223 1-617-577-1017 (Fax)	Maple V does symbolic mathematical manipulations. It comes with a library of mathematical functions and has its own Maple V programming language. It produces 2-D and 3-D graphs. Platform: Most UNIX, Macintosh, PC/Windows, VMS
Mathcad Plus 5.0	MathSoft, Inc. 101 Main Street Cambridge, MA 02142-1521 1-800-628-4223 1-617-577-1017 (Fax)	Mathcad Plus does technical calculations, including advanced numeric and symbolic calculations, differential equations, and matrix analysis. It will produce 2-D and 3-D graphs of the results. The user can enter and edit text on a Mathcad document, which has a spell checker. It has online help and a C and C++ programming interface. Platform: PC/Windows, Windows NT
Microsoft PowerPoint	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399 1-206-882-8080	PowerPoint is a tool for creating presentation slides. It provides an outline mode, a slide show mode, and various multimedia effects for powerful communication of ideas. Platform: PC/Windows, Macintosh

This page intentionally left blank.

APPENDIX G. SYSTEMS ENGINEERING TRENDS ASSESSMENT

G.1 ASSESSMENT STRATEGY

The assessment began by identifying key trends likely to impact the practice of systems engineering for software-intensive systems. Based upon the identified trends, the assessment sought to define the projected changes to the practice of systems engineering.

G.2 IDENTIFICATION OF KEY TRENDS IMPACTING THE PRACTICE OF SYSTEMS ENGINEERING

The motivation for the systems engineering trends assessment was the need to identify changes likely to occur during the next 6 to 8 years in the system engineering process as currently practiced in most large companies. The GSEP is compatible with the anticipated changes that were identified.

Process changes require many years to implement since they impact the culture of the organization and ways of doing business that have remained relatively unchanged for many years. There is significant value to projecting the types of process changes that will be needed in the next decade because of the lead-time required for systems engineering process improvement in any large company.

The major tasks undertaken in this activity were:

- Assessment of key trends likely to cause changes to the system engineering process for software-intensive systems
- Definition of the projected impacts of the trends on the systems engineering process

The trends analysis was performed by:

- Establishing a set of "strawman" trends and anticipated impacts based upon interviews with Consortium personnel working in the systems engineering area and the systems engineering TAG.
- Reviewing the "strawman" material with:
  - The Consortium Visiting Committee
  - Members of the Technical Advisory Board (TAB)
  - The Systems Engineering TAG

- Conducting telephone interviews with several staff members in Consortium-affiliated companies who were identified by members of the Consortium Board of Directors, the TAB, or the TAG as systems engineering "visionaries"

- Conducting a survey of members of the TAG regarding key issues raised by previous work on trends analysis

To ensure that the projected impacts on the systems engineering process are likely to have significant industrial relevance, the GSEP team sought to validate that the impacts are observable in some organizations today, although the impacts may not yet be widely felt.

The trends identified were categorized in the following manner:

- Government systems marketplace trends
- Product marketplace trends
- Corporate environment trends
- Technology trends

Sections G.2.1 through G.2.4 contain the most important trends identified in each area; the ones in bold were deemed to have the most significant potential impact on the practice of systems engineering based upon interviews with Consortium TAG members and key personnel in Consortium-affiliated companies.

G.2.1 Government Systems Marketplace Trends

- **More software-intensive systems**

  These systems are characterized by essential system functionality being provided by software, with complexity in both control and data.

- **Increased pressure to greatly reduce development cycle time (for increasingly complex systems)**
- More reengineering and evolution of existing systems
- More emphasis on the use of commercial products and practices in place of government-specific products and practices
- Growth in systems integration needs

G.2.2 Product Marketplace Trends

- **More complex computer-based products**

  These products are characterized by increasing software functionality, with distributed computing architectures becoming the norm.

- **Increased pace of product innovation and quality improvements**
- Increased competition in global markets
- More customers seeking quality-certified suppliers

G.2.3 Corporate Environment Trends

- Corporate downsizing

  Downsizing is characterized by:

  - Lower budget and reduced staff, but with higher management expectations
  - A smaller core work force and more use of contract services
  - A corporate organization with fewer levels of management

- More use of IPTs
- Increased international teaming and business alliances
- Realization that information is a fundamental resource that drives competitive advantage

G.2.4 Technology Trends

- Rapid changes in technology in all aspects of computer-intensive systems
- **Open systems from alternative suppliers**
- **Enhanced telecommunications capabilities**

  A particularly important aspect of this trend is the rapid growth of Internet services and the projected investment in the National Information Infrastructure.

- Visual, model-based techniques for simulating a system prior to construction
- Distributed computing environments supporting team development
- Flexible design and manufacturing to increase customization

G.3 PROJECTED IMPACTS ON THE PRACTICE OF SYSTEMS ENGINEERING

Sections G.3.1 through G.3.4 identify the projected changes based on the six trends identified as having the most significant impacts on the practice of systems engineering.

G.3.1 More Complex Software-Intensive Systems and Computer-Based Products

- Rapid model-based prototyping to involve customers and to validate requirements

- Products designed using a reuse-oriented systems engineering process
- Risk-management-driven processes
- ISO 9000 (International Standards Organization 1987) certified processes
- Separation of system integrators from "best in class" hardware and software component builders

G.3.2 Increased Pressure to Greatly Reduce Development Cycle Time (Government Marketplace) and Increased Pace of Product Innovation and Quality Improvements (Product Marketplace)

- Willingness to consider and accept existing solutions, which may not be the "best" or "optimal" solutions but are responsive and adequate ("good enough") approaches
- Separation of system integrators from "best in class" hardware and software component builders
- Rapid model-based prototyping to involve customers and to validate requirements
- Geographically distributed concurrent engineering
- Products designed using a reuse-oriented systems engineering process

G.3.3 Enhanced Telecommunications Capabilities

- Geographically distributed concurrent engineering
- Acceptance and use of modern information interchange techniques and elimination of adherence to hard-copy communications
- Use of computer-based systems engineering environments for both design and system understanding
- Electronic engineering information capture

G.3.4 Open Systems From Alternative Suppliers

- Willingness to consider and accept existing solutions, which may not be the "best" or "optimal" solutions but are responsive and adequate ("good enough") approaches.
- Performance-based system development will become common.

  In performance-based development, the customer only provides overall performance requirements, without specifications and standards.
- Separation of system integrators from "best in class" hardware and software component builders.

G.4 GSEP SUPPORT FOR FUTURE TRENDS

This section discusses the way that the GSEP supports or addresses these future trends. The following changes in the practice of systems engineering are projected to result from trends other than those listed in Section G.3:

- System design and implementation options will be increasingly constrained by legacy systems.
- A total life-cycle management emphasis in systems engineering will be used to increase reliability and maintainability.
- Integrated models will be used to assess system impacts during the design process.
- Use of more formally trained systems engineers.
- Clarifying, formalizing, and expediting the flow of information from systems engineering to the design and support disciplines.

G.4.1 Changes Related to Process Activities

The following future trends are projected to impact systems engineering process activities:

- Rapid model-based prototyping to involve customers and to validate requirements. The Evaluate Alternatives activity in the GSEP addresses the creation and analysis of these models (see Section 3.4.2.5).
- Products designed using a reuse-oriented systems engineering process. This report describes how GSEP can be used to engineer and apply reusable system parts (see Section 5).
- Risk-management-driven processes. Both the management and the technical processes in GSEP are risk driven. The management process uses risk factors as part of the decision making criteria (see Section 3.4.1). The technical process uses risk to evaluate and compare alternatives (see Section 3.4.2.5).
- Willingness to consider and accept existing solutions, which may not be the "best" or "optimal" solutions, but are responsive and adequate ("good enough") approaches. In the Evaluate Alternatives activity, the limitations of the alternative systems can be evaluated to determine if they meet the user's needs and are "good enough" (see Section 3.4.2.5).
- Integrated models will be used to assess system impacts during the design process. Integration of the subsystem or component system definitions has two separate aspects that must be considered (see Figure 62). First, there is the integration of the system design represented by the arrows on the left half on the figure. The evaluate activities use the design information to evaluate the integrated system at that level. This evaluation information is passed up to the next higher level and is used as part of the evaluation of the design done at that level. Second, there is the integration of the system implementation represented by the arrows on the right half on the figure. The implementation activities produce deliverable products that are tested to determine system characteristics such as system performance. The test results are passed up to the next higher level and are used as part of the testing at that level. The test results, at the

various levels, are also used as part of the design evaluations, and this is represented by the arrows that connect the test activities on the right of the figure to the evaluate activities on the left of the figure.

The process integration of the management and technical subprocesses in GSEP guarantees that the required process information is transmitted as part of evaluation/test of the "system" at each level (see Section 3.3).

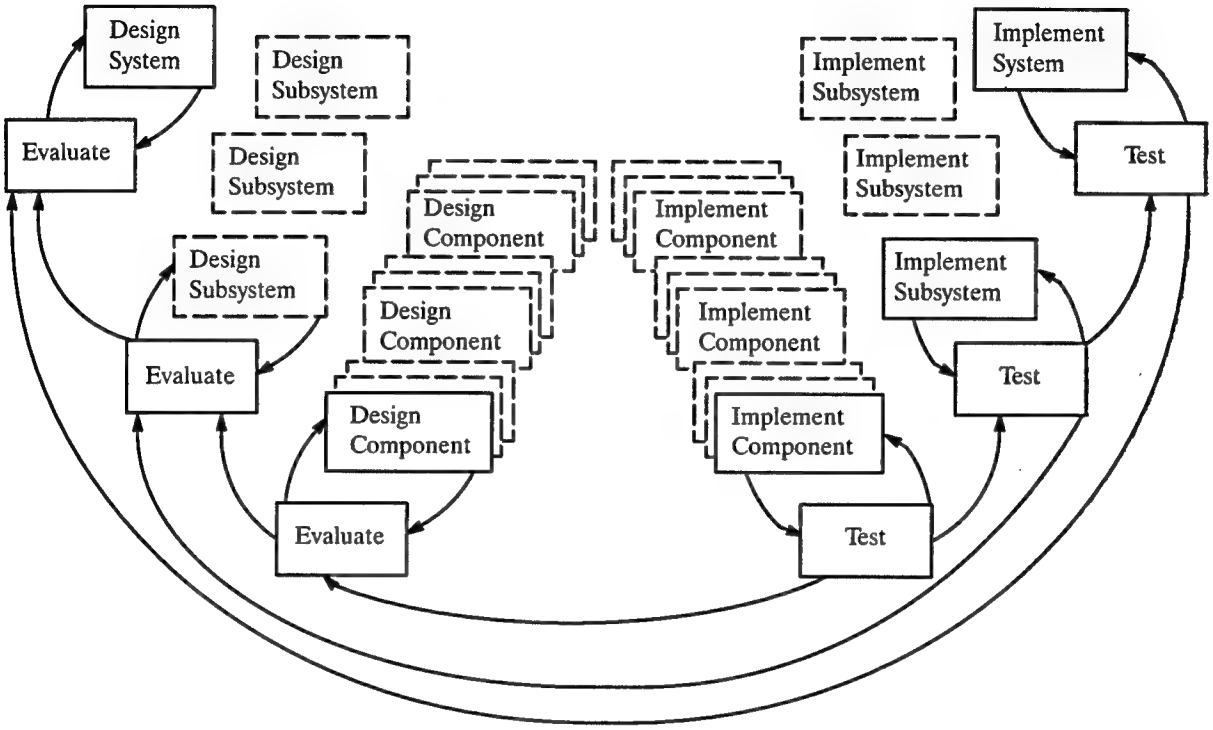


Figure 62. Product-Level Integration Across a System

- Clarifying, formalizing, and expediting the flow of information from systems engineering to the design and support disciplines. The GSEP addresses the flow of information from an engineered system to its subsystems and/or components (see Section 3.3.2).

G.4.2 Changes Related to Process Properties

The following future trends are projected to impact the properties exhibited by a systems engineering process:

- ISO 9000 compliant. GSEP defines a process that through process specialization can be made ISO 9000 compliant (see Section 4).

- Performance-based system development will become common (in performance-based development the customer only provides overall performance requirements, without specifications and standards). The GSEP's integration scheme (see Figure 62) can be used to ensure the implementation of any system characteristic such as system performance. In addition, the GSEP uses the concept of process tailoring to define a systems engineering

process that develops products that conform to whatever standards are currently available and appropriate for the development effort (see Section 4.2.1.2).

- System design and implementation options will be increasingly constrained by legacy systems. The GSEP is appropriate for engineering and reengineering systems, especially systems that are complex or are being constructed using already existing elements (see Section 3.3.2).

- A total life-cycle management emphasis in systems engineering will be used to increase reliability and maintainability. The GSEP addresses life-cycle management by the introduction of a life-cycle process architecture. This life-cycle process architecture is managed as a system and developed using the technical activities in the GSEP (see Section 4.2.2.1).

G.4.3 Changes Related to Process Support Tools, Roles, and Responsibilities

The following future trends are projected to impact the selection of tools, roles, and responsibilities that are defined to support a systems engineering process:

- Use of computer-based systems engineering environments for both design and system understanding. The GSEP process is well-defined and, thus, appropriate for automation. (See Appendix F for a discussion of automated support for the GSEP.)

- Separation of system integrators from "best in class" hardware and software component builders. This trend is not addressed in the GSEP.

- Geographically distributed concurrent engineering. The GSEP supports/addresses concurrent engineering (see Section 3.3.3). The issues with a distributed tooling environment are discussed in Appendix F.

- Acceptance and use of modern information interchange techniques and elimination of adherence to hard-copy communications. This trend is not addressed in the GSEP.

- Electronic engineering information capture. See Appendix F for tooling issues in capturing the engineering data.

This page intentionally left blank.

LIST OF ABBREVIATIONS AND ACRONYMS

CALS	Computer-Aided Logistics Support
CBS	cost breakdown structure
CGM	Computer Graphics Metafile
COTS	commercial off-the-shelf
CPI	Continuous Process Improvement
DoD	Department of Defense
EoS	Estimate of the Situation
ESP	Evolutionary Spiral Process
GSEP	Generic Systems Engineering Process
ICAM	integrated computer-aided manufacturing
IDEF	ICAM Definition
IPT	integrated product team
NCOSE	National Council on Systems Engineering
OLE	Object Linking and Embedding
QFD	Quality Function Deployment
RMP	Risk Management Plan
SE-CMM	Systems Engineering Capability Maturity Model
SEI	Software Engineering Institute
SOW	Statement of Work
SGML	Standard Generalized Markup Language
TAB	Technical Advisory Board
TAG	Technical Advisory Group

TPM	technical performance measures
V & V	verification and validation
WBS	work breakdown structure

REFERENCES

Blanchard, B. 1991 — *System Engineering Management*. New York, New York: John Wiley.

Blanchard, B.S., W. Fabrycky, and D. Verma, eds. 1994 — *Application of the System Engineering Process to Define Requirements for Computer-Based Design Tools*. New Carrollton, Maryland: The Society of Logistics Engineers.

Blanchard, B.S., and W.J. Fabrycky 1981 — *System Engineering*. Englewood Cliffs, New Jersey: Prentice-Hall.

1990 — *System Engineering*. 2d ed. Englewood Cliffs, New Jersey: Prentice-Hall.

Boehm, Barry W., and Rony Ross 1989 — "Theory W Software Project Management: Principles and Examples." In *Tutorial: Software Risk Management*. Washington, D.C.: IEEE Computer Society Press.

Campbell, G., S. Faulk, and D. Weiss 1990 — *Introduction to Synthesis,* INTRO_SYNTHESIS_PROCESS-90019-N. Herndon, Virginia: Software Productivity Consortium.

Chapman, W.L., T. Bayhill, and A.W. Wymore 1992 — *Engineering Modeling to Design*. Boca Raton, Florida: CRL Press.

Charette, Robert N. 1990 — *Applications Strategies for Risk Analysis*. New York, New York: Intertext Publications, McGraw-Hill.

Chase, W.P. 1974 — *Management of System Engineering*. New York, New York: John Wiley.

Chestnut, H. 1965 — *Systems Engineering Tools*. New York, New York: John Wiley.

1967 — *Systems Engineering Methods*. New York, New York: John Wiley.

Department of Defense 1974 — *Engineering Management*. MIL-STD-499A (May). Washington, D.C.: Department of Defense, U.S. Air Force.

1994 — *System Engineering* (For Coordination Review) MIL-STD-499B (May 6). Dayton, Ohio: Aeronautical Systems Division (ASD/ENS) Wright Patterson Air Force Base.

Electronics Industry Association 1994 — *DRAFT EIA Interim Standard 632 Systems Engineering.* Washington, D.C.: Electronics Industry Association 20006–1813 (December).

Eisner, H. 1988 — *Computer-Aided Systems Engineering.* Englewood Cliffs, New Jersey: Prentice-Hall.

Goode, H., and R.E. Machol 1957 — *Systems Engineering: An Introduction to the Design of Large Scale Systems*. New York, New York: McGraw-Hill.

Hall, A.D. 1962 — *Methodology for Systems Engineering.* Princeton, New Jersey: Von Nostrand.

IEEE 1993 — *Standard for Systems Engineering* (DRAFT Rev 1.0, December). New York, New York, IEEE P1220.

International Standards Organization 1987 — *Quality Management and Quality Assurance Standards.* International Standards Organization.

Kerzner, Harold 1984 — *Project Management: A Systems Approach to Planning, Scheduling and Controlling*. New York, New York: Van Nostrand Reinhold Co.

Kuhn, D.A. 1993 — "Automating Systems Engineering." Master's thesis, The University of Texas at Dallas.

Lacy, J.A. 1992 — *Systems Engineering Management Achieving Total Quality*. New York, New York: McGraw-Hill.

Pohlmann, L.D. 1993 — "Systems Engineering Practices Committee Activities: Plenary Session Reports." In *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering*, July 26–28. Arlington, Virginia.

Rechtin, E. 1991 — *Systems Architecting Creating and Building Complex Systems.* Englewood Cliffs, New Jersey: Prentice-Hall.

Reilly, Norman B. 1993 — *Successful Systems Engineering for Engineers and Managers*. New York, New York: Van Nostrand Reinhold.

Royce, Winston W. 1970 — "Managing the Development of Large Software Systems." In *Proceedings, IEEE WESCON.*

Sage, A.P. 1977 — *Methodology for Large Scale Systems*. New York, New York: McGraw-Hill.

1992 — *Systems Engineering*. New York, New York: John Wiley.

Software Engineering Institute 1994 — *SE-CMM Model Description Systems Engineering Improvement.* SECMM-94-04, Release 2.03, November 8. Pittsburgh, Pennsylvania: Software Engineering Institute.

Software Productivity Consortium 1992 — *Software Measurement Guidebook*, SPC-91060-CMC, version 02.00.02. Herndon, Virginia: Software Productivity Consortium.

1993a — *Reuse-Driven Software Processes Guidebook*, SPC-92019-CMC, version 02.00.03. Herndon, Virginia: Software Productivity Consortium.

1993b — *System Engineering Workshop Summary*, SPC-93128-MC, version 01.00.05. Herndon, Virginia: Software Productivity Consortium.

1993c — *Using New Technologies: A Technology Transfer Guidebook*, SPC-92046-CMC, version 02.00.08. Herndon, Virginia: Software Productivity Consortium.

1994a — *Guest Lecture Documentary Series, Computer-Aided Systems Engineering*, SPC-94037-CMC, presented by Dr. Stephen J. Andriole. Herndon, Virginia: Software Productivity Consortium.

1994b — *Process Definition and Modeling Guidebook*, SPC-92041-CMC, version 02.00.02. Volumes 1 and 2. Herndon, Virginia: Software Productivity Consortium.

1994c — *Process Engineering With the Evolutionary Spiral Process Model*, SPC-93098-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.

Software Technology Support Center 1993 — *Draft Process Technologies Method and Tool Report.* Ogden, Utah: Software Technology Support Center, Ogden Air Logistics Center.

U.S. Air Force 1981 — *Integrated Computer-Aided Manufacturing (ICAM) Architecture.* Part II, Volume IV, *Function Modeling Manuel (IDEF$_0$).* AFWAL-TR-81-4023.

1988 — *Acquisition Management: Software Risk Abatement,* AFSC/AFLCP 800-45. Washington, D.C.: Andrews Air Force Base, Air Force Systems Command and Air Force Logistics Command.

This page intentionally left blank.

BIBLIOGRAPHY

Deming, Edwards E. *Out of the Crisis*. Cambridge, Massachusetts: MIT Press, 1986.

Department of Defense. *Defense Management College: Systems Engineering Management Guide*. Washington D.C.: U.S. Government Printing Office, 1990.

Sailor, J.D. "Systems Engineering: An Introduction." In *System and Software Requirements Engineering,* edited by R.H. Thayer and M. Dorfman. Los Altos, California: IEEE Computer Society Press, 1990.

Stephanou, S.E., Obradovitch, M.M. *Project Management, System Development and Productivity*. Malibu, C.A.: Daniel Spencer Publishing, 1985.

Thuesen, G.J., Fabrycky, W.J. *Engineering Economy*, Seventh Edition. Englewood Cliffs, N.J.: Prentice Hall, 1989.

Wymore, Wayne A. *Systems Engineering Methodology for Interdisciplinary Teams*. New York: John Wiley & Sons, Inc., 1992.

This page intentionally left blank.